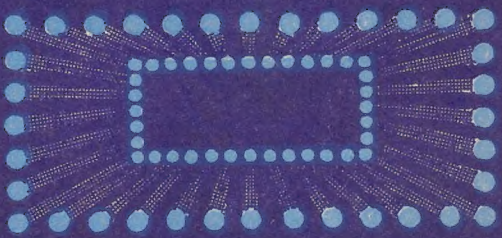
Г. Б. УИЛЬЯМС

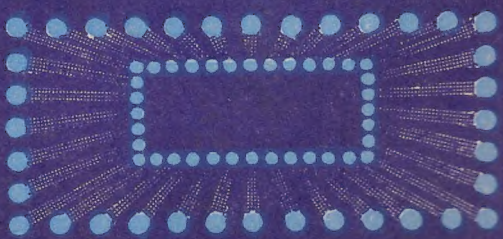
# Отладка микропроцессорных систем



Отладка микропроцессорных систем

Г. Б. УИЛЬЯМС

# Отладка микропроцессорных систем



Г. Б. УИЛЬЯМС

# Отладка микропроцессорных систем

Перевод с английского *В. Л. Григорьева*

Под редакцией *В. В. Сташина*

**Scan Pirat**

МОСКВА
ЭНЕРГОАТОМИЗДАТ
1988

ББК 32.973
У 36
УДК 681.31—181.48

Рецензент В. Л. Григорьев

WILLIAMS G. B.
**TROUBLESHOOTING ON MICROPROCESSOR BASED SYSTEMS**
Pergamon Rress Ltd., Oxford, 1984

**Уильямс Г. Б.**

У 36 Отладка микропроцессорных систем: Пер. с англ. — М.: Энергоатомиздат, 1988. — 253 с.: ил.

ISBN 5-283-02484-9

Рассмотрены практические вопросы отладки микропроцессорных систем и применяемое для этого контрольно-измерительное оборудование. Приведено описание логических и сигнатурных анализаторов. Показаны некоторые схемотехнические решения и программное обеспечение, необходимые при отладке микросистем.

Для инженерно-технических работников в области вычислительной техники и автоматики, занятых проектированием, производством и эксплуатацией микросистем.

У $\frac{2405000000\text{-}083}{051(01)\text{-}88}$ 284-87 **ББК 32.973**

ISBN 5-283-02484-9 (русск.)
ISBN 0-08-029988-1 (англ.)

© Pergamon Rress Ltd., 1984
© Энергоатомиздат, 1988 г.

# Предисловие к русскому изданию

В настоящее время не приходится много говорить о достижениях и роли микроэлектроники и микропроцессорной техники, поскольку их значение в народном хозяйстве общеизвестно. Для многих инженерно-технических работников микропроцессоры стали основной сферой их производственной деятельности, и в обозримом будущем число таких специалистов будет возрастать. Кроме того, вопросы проектирования микропроцессорных систем вошли в учебные планы многих специальностей высших технических учебных заведений. Именно на инженерно-технических работников и студентов институтов, специализирующихся в области вычислительной техники, рассчитана настоящая книга.

Монография Г. Б. Уильямса, по существу, является первым изданием, целиком посвященным вопросам отладки микропроцессорных систем, и она, безусловно, вызовет интерес советских читателей. Не следует искать в книге обсуждения теоретических аспектов отладки микропроцессорных систем. Книга написана инженером и для инженеров. В ней сосредоточены личный опыт автора и рекомендации его коллег, поэтому мотив практической направленности рассматриваемых вопросов отчетливо звучит на протяжении всей книги.

Мы не будем касаться обзора содержания книги, так как он достаточно подробно изложен в предисловии автора. Отметим только, что книга написана четким «техническим» языком, хотя иногда автор приводит и общие сведения. Очевидно, наибольший интерес читателей вызовут главы, посвященные логическим и сигнатурным анализаторам, т. е. приборам, разработанным специально для отладки микропроцессорных систем.

В иллюстративных примерах автор большей частью опирается на 8-битный микропроцессор Z80 фирмы Zilog, США. Архитектурно этот микропроцессор напоминает отечественный микропроцессор КР580ИК80, но имеет два набора внутренних регистров и несколько расширенную систему команд. Кроме того, в этих микро-

процессорах применяются различные мнемоники ассемблерных команд, например LD и MOV, DEC и DCR, INC и INR. Однако разобраться в простых фрагментах программ несложно, а расширенные возможности микропроцессора Z80 почти не привлекаются.

Приходится сделать традиционное замечание об используемой терминологии. Русский и английский языки довольно богаты разнообразными терминами, относящимися к микропроцессорной технике и, в частности, отладке микропроцессорных систем, а общий терминологический стандарт отсутствует. При переводе мы придерживались следующих основных терминов: «поиск неисправностей» (troubleshooting, fault finding), «тестирование» (testing), «контроль» (Checking), «отказ» (fault), «неисправность» (failure, malfuntion), «тест» (test), и сделали исключение только в названии книги. Кроме того, введены такие термины, как «пульсатор» (pulser), «пробник» (probe) и «индикатор тока» (current tracer). Будем надеяться, что читатели сами сориентируются в тонкостях трактовки приведенных терминов.

Конечно, у каждого читателя сложится свое впечатление о прочитанной книге и ее значении в профессиональной деятельности. Нам остается ожидать, что большинство читателей воспримут книгу с интересом.

*В. Л. Григорьев*

# Предисловие

Микропроцессорая революция длится уже более десятилетия; за это время применяемые приборы прошли путь от простейших вычислительных элементов до сложных многофункциональных подсистем, реализованных в одной микросхеме. В истории исключительно быстрого развития микроэлектроники были драматические и довольно острые моменты. Многие приборы, разработанные за прошедшее время, были отвергнуты рынком, а другие, наоборот, получили очень широкое распространение и стали промышленными стандартами. Мы находимся на этапе создания микросистем пятого поколения, с которыми связывается появление компьютерных архитектур, радикально отличающихся от классической архитектуры Джона фон Неймана.

Но, как это часто бывает с быстро прогрессирующими технологиями, промышленность, в общем, медленно адаптируется к достижениям микроэлектроники, что объясняется разнообразными причинами. В промышленности действует естественный консерватизм, особенно когда дело касается замены оборудования, а старая технология дает неплохие результаты. Появление большого числа комплектов приборов, выполняющих примерно одинаковые функции, ставит проблему выбора подходящего комплекта. Работающие в промышленности инженеры и техники должны изучать все новые и новые системы, на что уходит много времени и расходуются значительные средства. Все эти факторы приводят к тому, что от момента разработки изделия до его широкого применения проходят два-три года.

В отличие от консервативного подхода в промышленности новые фирмы оказываются заинтересованными во внедрении современных технологий. Стремительный рост рынка персональных компьютеров служит ярким примером сферы сбыта, возникшей благодаря достижениям микроэлектроники. Фирмы оказались заинтересованными в разработке законченных вычислительных систем как средства увеличения продажи их продукции.

Структура вычислительной системы ставит уникальные проблемы в случае ее отказа и проведения тестирования. К сожалению, разработка необходимой контрольно-измерительной аппаратуры заметно отстает от развития собственно микроэлектронных приборов. В первых вариантах контрольно-измерительной аппаратуры использовались традиционные приборы, рассчитанные на восприятие и отображение информации от микропроцессорных систем. Индицируемую информацию было трудно интерпретировать, а настройка приборов была довольно сложной. Со временем в разработке контрольно-измерительной аппаратуры стали учитываться требования эргономики с целью упрощения работы с ней и удобного представления результатов.

Сами способы поиска неисправностей в микросистемах развились от возложения большинства забот на пользователей до встраивания средств контроля в сами системы с соответствующей «разгрузкой» пользователей. При многих методах контроля микропроцессорных систем требуются лишь незначительные дополнительные средства, которые проще встроить при разработке и производстве системы, чем дорабатывать готовое изделие.

Целенаправленные проверка и ремонт неисправной системы предполагают знание персоналом конкретной системы, умение правильно выбрать необходимую аппаратуру и наличие тщательно продуманного плана работ. Мы не будем касаться первого требования, а сосредоточим внимание на втором и третьем требованиях. Продуманный подход к проверке любой системы можно представить в виде так называемого «дерева поиска неисправностей», которое превращает общую задачу проверки в последовательность действий; на каждом этапе этой задачи учитываются результаты предыдущих действий. В предлагаемой вниманию читателей книге раскрывается данный подход как наиболее результативный для цифровых систем и показывается, какие контрольно-измерительные приборы следует применять в конкретных ситуациях.

Книга ориентирована на всех, кто желает познакомиться с отладкой микропроцессорных систем и применяемой для этого аппаратурой. Ее материал представит интерес для инженеров и техников, которые хотят расширить свои знания в быстро прогрессирующей микро-

процессорной технике. Книга написана таким образом, что каждая из ее глав посвящена отдельным вопросам и сравнительно автономна. Однако добиться полной независимости отдельных глав не удалось.

Значительная часть книги написана по материалам лекций, прочитанных автором на курсах повышения квалификации инженеров и аспирантов.

Вводная глава по микропроцессорным системам включена только ради полноты охвата материала и не претендует на всеобъемлющее изложение вопроса. Основное внимание здесь сосредоточено на таких вопросах, как тристабильные элементы и дешифрирование адреса, которые очень важны для понимания работы вычислительных систем, но излагаются в книгах довольно бегло. Приведена схема микроконтроллера, являющегося управляющим элементом во многих производственных процессах, а в конце главы кратко рассмотрены вопросы программирования.

В главе 2 содержится обзор тех проблем, которые характерны для систем с шинной структурой, и обсуждаются основные принципы проверки и локализации отказов в компьютерах. Система рассматривается как ядро, окруженное периферийными микросхемами, и для проведения основных проверок ядро должно функционировать.

Глава 3 посвящена принципам контроля систем, особенно в условиях опытного производства. Рассмотрены виды специальных воздействий и «дерево поиска неисправностей» как основа локализации отказов.

Глава 4 касается применения для контроля микрокомпьютеров таких традиционных контрольно-измерительных приборов, как осциллографы и цифровые вольтметры. Основное внимание обращено на их ограничения и необходимость разработки специальных приборов, предназначенных для анализа отказов в микрокомпьютерах.

В гл. 5—8 описываются приборы, ориентированные на цифровые системы. В гл. 5 речь идет о таких простейших приборах, как логические пробники, логические пульсаторы, индикаторы тока и логические компараторы. Дано описание принципов их работы и способов применения в типичных ситуациях. Анализ ограничений этих простых средств служит введением для последующих

глав, в которых рассматриваются более сложные приборы.

В гл. 6 обсуждаются логические анализаторы и их эволюция от сравнительно простых приборов на основе осциллографов до сложнейших современных моделей. Попутно рассмотрены виды индикаторов и их применение. Несколько идей, изложенных в этой главе, разработано слушателями курсов, за что автор выражает признательность П. Девису и С. Хантеру.

Материал гл. 7 посвящен сигнатурному анализу и его происхождению из способа подсчета переходов. Дано математическое обоснование сигнатурного анализа и обеспечиваемой им высокой вероятности обнаружения ошибок. Необходимо, чтобы проверяемая система была рассчитана на применение сигнатурного анализа, а для этого требуются дополнительные схемы и встроенные тест-программы. Тем не менее большинство существующих систем можно приспособить для сигнатурного анализа.

Глава 8 знакомит читателей с таким способом проверки, как эмуляция. Она начинается с обсуждения моделирования как средства имитации поведения системы и заканчивается методами управления одной системой от другой таким образом, что возможно реализовать различные уровни эмуляции. Термин «эмуляция» используется сейчас применительно к разнообразным приборам, начиная большими системами проектирования и кончая автономными контрольными приборами; в книге обсуждаются обе эти крайности. Эмуляционное контрольное оборудование опирается на использование многих из ранее рассмотренных методов и характеризуется широким спектром возможностей приборов.

Глава 9 содержит сведения по видам тест-программ, предназначенных для контроля компонентов системы. Такие тест-программы варьируются от программ самоконтроля, которые выполняются при включении системы, до программ, инициируемых только после возникновения отказа. Приведены примеры тест-программ, которые можно использовать для контроля основных компонентов вычислительной системы.

В последней главе затронуты вопросы контроля периферийного оборудования вычислительных систем, рассмотрен функциональный контроль системы на примере системы сбора данных, конфигурация которой рас-

считана на этот вид контроля, описано использование индикаторов «проходит/не проходит», которые упрощают эксплуатацию систем, приведены сведения о стандарте последовательной связи RS232C и о шине параллельного интерфейса IEEE-488.

Главное назначение книги — осветить основные современные методы контроля и показать те принципы, на которых они базируются. По мере усложнения систем и разработки новых методов контроля совершенствуется и контрольно-измерительная аппаратура. Поэтому материал книги актуален только для современного уровня самих систем и приборов и со временем, конечно, должен видоизмениться. Тем не менее книга может служить отправной точкой и должна дать читателю такую основу, которая будет применима и к последующим разработкам.

Выражаю признательность П. А. И. Девису и К. Р. Веберу за внимательное изучение рукописи и полезные предложения, которые способствовали улучшению материала книги и формы его изложения.

*Г. Б. Уильямс*

## Список условных обозначений

$T$ — транзистор
$D$ — диод
$R$ — резистор
$C$ — конденсатор (емкость)
$V_{cc}$ — максимальное положительное напряжение в биполярной схеме (обычно + 5 В)
$V_{dd}$ — максимальное положительное напряжение в КМОП-схеме (обычно + 5 В)
$V_{ss}$ — максимальное отрицательное напряжение в КМОП-схеме (обычно потенциал земли)
$V_{IL}$ — входное напряжение логического элемента в состоянии логического «0»
$V_{IH}$ — входное напряжение логического элемента в состоянии логической «1»
$V_{OL}$ — выходное напряжение логического элемента в состоянии логического «0»
$V_{OH}$ — выходное напряжение логического элемента в состоянии логической «1»
$V_N$ — напряжение шумов (помех)
$N$ — число последовательностей
$P_x$ — вероятность $x$

# 1. Введение в микропроцессорные системы

Вряд ли найдутся электронные компоненты, которые могут соперничать с микропроцессорами по темпам развития и разнообразию применений. Первый микропроцессор фирма Intel разработала в 1971 г., а вскоре появилось множество приборов других фирм. В микропроцессоре сделана попытка реализовать в одной микросхеме основные функции *центрального процессора* (ЦП) компьютера. Микропроцессоры первого поколения имели элементарную архитектуру по сравнению с типичным миникомпьютером, который был в то время самым маленьким и дешевым цифровым компьютером. За первыми микропроцессорами последовали микропроцессоры второго и третьего поколений, которые стали гораздо мощнее и сложнее своих предшественников. Увеличение плотности упаковки и повышение быстродействия микропроцессоров прямо связаны с совершеиствованием технологии производства интегральных микросхем.

Появлеиие сложных функциональных устройств в виде отдельных микросхем привело к смещению разработок из области схемотехники в область системотехники. Компоненты системы обладают всеми свойствами компьютера, а функциональные характеристики системы определяются совокупностью программ. По существу, разработчик настраивает универсальный набор компонентов на выполнение конкретных функций системы, заставляя компьютер выполнять последовательиость команд, реализующих требуемые действия. Поведением системы управляет ее программное обеспечение, т. е. совокупность программ.

Вычислительная система представляет собой симбиоз электрических компонентов (или аппаратных средств) и программ (или программного обеспечения), которые тесно взаимосвязаны и практически бесполезны друг без друга. Такой симбиоз в случае неисправности изделий с микропроцессорами ставит довольно трудную проблему определения того, где возникла неисправность — в аппаратных средствах или в программном

обеспечении. Помочь в решении подобного рода проблем призвана специальная контрольно-измерительная аппаратура, номенклатура которой варьируется от простых приборов до сложных, многофункциональных систем.

Инженеры и техники, занятые разработкой, эксплуатацией и ремонтом вычислительных систем, должны иметь теперь более высокий уровень технической культуры по сравнению с их традиционным уровнем. Необходимо совершенствовать систему образования, чтобы справиться с этими жизненными потребностями, особенно в условиях непрерывной эволюции основных принципов проектирования и контроля вычислительных систем. Широкое распространение компьютеров поставило вопрос о разработке системного программного обеспечения на всех уровнях программирования. Проблемы, возникающие при объединении процессоров в распределенную систему, и проблемы управления такими системами пока окончательно не решены. Контроль и проверка сложных систем с помощью встроенных тест-программ и специальной аппаратуры серьезно отстают от масштабов применения таких систем.

### 1.1. Компьютер

В составе любого компьютера имеются три основных устройства, показанных на рис. 1.1.

*Центральный процессор* (ЦП) управляет компьютером, а также выполняет все арифметические и логические операции. В микрокомпьютере функции ЦП реализует микропроцессор.

*Память* (или запоминающее устройство) хранит последовательности команд программы, которые выполняет ЦП. Процессор последовательно считывает каждую команду, дешифрирует ее и производит операции, закодированные в комаиде. Последовательное выполнение команд центральным процессором определяет действия системы. В памяти хранятся также даиные, например, промежуточные результаты вычислений, которые неудобно хранить в ЦП.

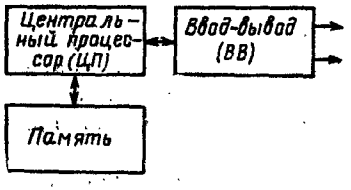



**Рис. 1.1.** Состав технических средств компьютера

Средства *ввода-вывода* (ВВ) обеспечивают взаимодействие компьютера с внешним миром. Под управлением ЦП данные считываются через входиой порт или передаются в какое-либо внешнее устройство через выходной порт.

Типы подключаемых к компьютеру внешних устройств определяются назначением системы. В компьютерах общего назначения применяются такие периферийные устройства, как клавиатура, видеодисплей и принтер, которые обеспечивают взаимодействие человека с машиной. К управляющим цифровым компьютерам подключаются преобразователи, дающие электрические аналоги таких переменных, как давление, температура и скорость потока. Выходные порты соединяются с исполнительными устройствами, например с шаговым двигателем, для регулирования параметров процесса. Участие обслуживающего персонала в управляющих системах часто сводится только к коррекции установочных значений параметров.

Компьютер представляет собой универсальную структуру, которую путем подключения соответствующих внешних устройств и программирования можно использовать для решения разнообразных задач.

### 1.2. Системы с шинной структурой

Информация передается из одного блока вычислительной системы в другой по линиям связи в форме двоичиых (двузначных) сигиалов. Такие сигналы имеют только два допустимых состояния или уровня напряжения, которые иазываются состояниями логического 0 и логической 1. Установлено соглашение о том, что иуль представляет состояние логического 0 и +5 В — состояние логической 1 (так называемая позитивная логика).

Если информация передается из одного места в другое по одной линии путем представления логических уровней в виде последовательиых временных рядов, то такой вид связи иазывается последовательной передачей.

На рис. 1.2 показана временна́я диаграмма пересылки 8-битиого двоичного слова 01001100 по линии последовательной передачи. Первым в показанном слове передается *старший бит;* на практике прииято передавать

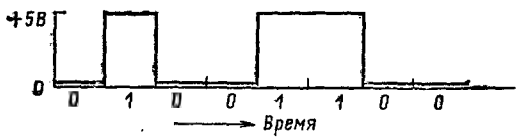



**Рис. 1.2.** Последовательная передача

слово данных в обратном порядке, т. е. начиная с *младшего бита*. В последовательной передаче применяется минимальное число линий связи, но для передачи слова из $N$ бит требуется $N$ тактов синхронизации. При другом виде передачи каждый из $N$ бит посылается по отдельной линии, а сами линии упорядочиваются в соответствии с двоичными весами. В результате получается параллельная передача, а совокупность линий, по которым передается информация, называется *шиной*.

Информация передается по шине параллельно битами и последовательно словами, причем передача каждого слова длится всего один такт синхронизации. Каждый элемент данных считывается с шины синхронно с системной синхронизацией. На рис. 1.3 показана передача слова 01001100 по 8-битной шине в некоторый момент времени $t$.

Для достижения высокой скорости передачи данных в компьютере применяются шины. Конечно, система с шинной структурой усложняет компоненты компьютера, но это компенсируется повышением его производительности.

Как показано на рис. 1.4, в компьютере передача одного элемента данных из одного блока в другой осуществляется с использованием трех шин.

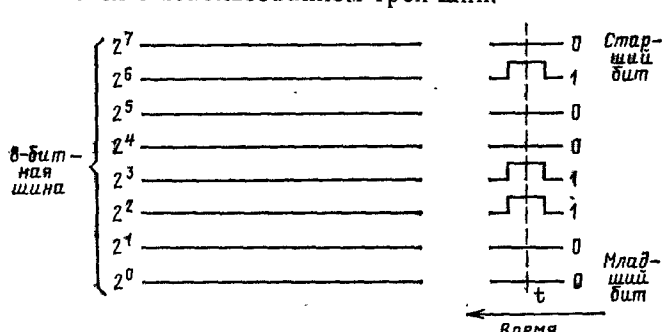



**Рис. 1.3.** Параллельная передача по шине

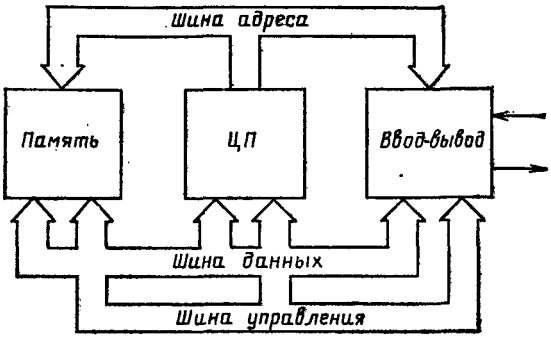



**Рис. 1.4.** Компьютер с шинной структурой

Передачами данных управляет ЦП, который определяет источник или получатель данных, помещая его адрес на *шину адреса*. В простых системах выдавать адреса может только ЦП, поэтому шина адреса называется однонаправленной. Число линий или ширина шины адреса определяет общее число ячеек, которое может адресовать ЦП. В большинстве микропроцессоров шина адреса имеет 16 линий, что позволяет ЦП выбирать любую из $2^{16}$ (65536) ячеек. Так как принято обозначать число $2^{10}$ (1024) буквой К, то диапазон прямой адресации (адресное пространство) такой системы составляет 64К.

Поместив код адреса на шину адреса, ЦП должен информировать остальные компоненты системы о типе инициируемой передачи. Операцией может быть считывание или запись, а участвовать в ней могут память и входные или выходные порты. ЦП определяет операцию сигналами в линиях *шины управления*, которые выбирают память или ввод-вывод и указывают операцию считывания или записи. Не все линии в шине управления являются для ЦП выходными; например, при операции считывания из памяти с большим временем доступа устройство памяти должно передать в ЦП управляющий сигнал, информирующий его о необходимости ожидания готовности запрошенных данных. Таким образом, одни сигналы в шине управления ЦП выдает, а другие принимает.

Определив с помощью шин адреса и управления адрес в системе и тип операции, ЦП либо посылает данные в

указанную ячейку, либо принимает данные из ячейки по *шине данных*. Любой из компонентов вычислительной системы, показанной на рис. 1.4, может передавать или принимать данные с шины данных, которая является двунаправленной. Число линий или ширина шины данных позволяет классифицировать компьютеры. Большинство микропроцессоров имеют 8 линий в их шинах данных и называются 8-битными микропроцессорами; некоторые микропроцессоры и большинство компьютеров имеют 16 линий в шинах данных и называются 16-битными микропроцессорами или 16-битными компьютерами.

Совокупность из 8 бит называется байтом, а ширина шины данных определяет формат слова компьютера. В 8-битном микропроцессоре по шине даиных передаются байты. Формат слова компьютера определяет также разрядность каждой из ячеек памяти и ширину его входных и выходных портов.

### 1.3. Системы с отображением ввода-вывода на память

В компьютере с шинной структурой, приведенном на рис. 1.4, средства ввода-вывода полиостью отделены от памяти системы. Большинство же миникомпьютеров и многие микрокомпьютеры считают входные и выходные порты просто ячейками памяти. Что касается ЦП, считывание из входного порта осуществляется точно так же, как будто ЦП считывает содержимое ячейки памяти. Аналогичным образом запись в выходной порт ЦП считает записью в ячейку памяти. Достоинство ВВ, отображенного на память, заключается в сокращении числа управляющих сигналов в шине управлеиия, так как ие нужен специальный сигнал, идентифицирующий операцию ВВ. Поскольку все передачи данных считаются обращениями к памяти, можно исключить и управляющий сигнал, указывающий обращение к памяти. Недостаток ВВ, отображенного на память, заключается в невозможности использовать занимаемые входными и выходными портами адреса для памяти и разделении всего адресного пространства системы между собственно памятью и вводом-выводом.

Альтернативой ВВ, отображенного иа память, является неуклюже названное ВВ, отображенное на ВВ, ко-

торое используется фирмами Intel и Zilog и где имеются отдельные управляющие сигналы для обращения к ячейкам памяти и портам ВВ. В микропроцессорах с данным способом ВВ шина управления несколько сложиее, а в системах команд предусматриваются специальные команды операций ВВ, которые, в общем, обладают меньшими возможностями, чем комаиды с обращением к ячейкам памяти. Достоиисство ВВ, отображенного на ВВ, заключается в полном разделении адресных пространств системной памяти и портов ВВ, что позволяет использовать для запоминающих устройств все адресиое пространство системы.

В архитектуре микропроцессоров, имеющих рассмотреиные способы отображения ВВ, есть различия. Микропроцессоры с ВВ, отображеиным на ВВ, имеют относительно большее число внутренних регистров для временного хранения данных, а микропроцессоры с ВВ, отображенным на память, имеют меньшее число регистров, что требует запоминания почти всех данных в памяти, внешней по отношению к ЦП.

### 1.4. Тристабильные схемы

Каждый из функциональных блоков в вычислительной системе с шинной структурой может помещать данные на шиу данных по запросу от ЦП. Блоки подключаются к шине данных с помощью логических элементов, которые в их обычном исполнении могут выводить на свои лиии шины данных только уровни логического 0 и логической 1. К каждой линии шины данных подключаются выходы множества логических элементов, одна часть которых может иметь один логический уровень, а другая часть — другой уровень. В этих условиях фактическое состояние линии оказывается неопределенным, и возникает так называемый конфликт на шине, когда одно из выходных устройств пытается установить на линии один уровень, а другое устройство — другой уровень.

Основной структурой логического элемента, которая обеспечивает активный источник тока на выходе и активный отвод тока с выхода, является ТТЛ-элемент НЕ-И7400, представленный на рис. 1.5. В выходном каскаде элемента 7400 один из транзисторов ($T_3$, $T_4$) всегда включен, а другой выключен. Если этот элемент исполь-

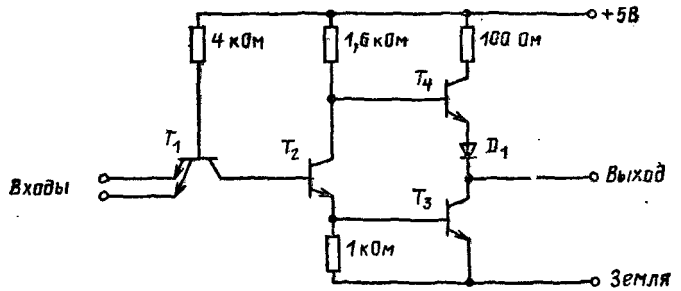



Рис. 1.5. ТТЛ-элемент НЕ-И 7400

зовать в качестве драйвера линии шины данных, конфликты на шине неизбежны. Однако схему элемента можно модифицировать путем введения двух дополнительных компонеитов, что приведет к появлению элемента с тремя возможными выходными состояниями.

В схему элемента 7400 вводится диод $D_2$, а верхний транзистор выходного каскада превращается в схему Дарлингтона путем добавления транзистора $T_5$, как это показано на рис. 1.6. Когда управляющий вход $C$ подключается на землю (логический 0), диод $D_2$ фиксирует потенциал коллектора транзистора $T_2$ примерно иа уровне 0,6 В с учетом прямого смещения диода $D_2$. Один из переходов база — эмиттер многоэмиттерного транзистора $T_1$ смещен в прямом направлении, что вызывает включение транзистора $T_1$, и его коллекторный потенциал оказывается ниже уровня, необходимого для поддержания $T_2$ во включенном состоянии. Транзистор $T_2$ выключается, его эмиттерный ток прекращается, и, сле-

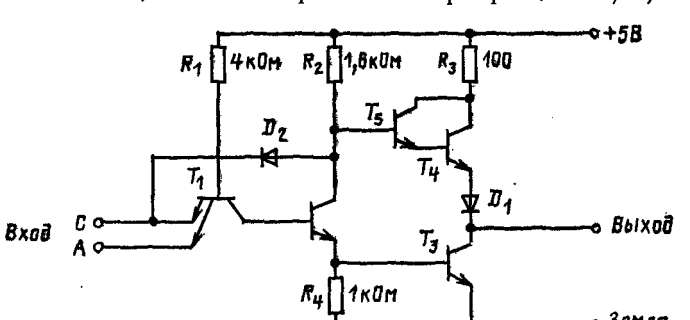



Рис. 1.6. Тристабильный элемент (драйвер)

довательно, падение напряжения на резисторе $R_4$ равно нулю. Потенциал на коллекторе $T_2$ недостаточен для включения транзисторов $T_5$ и $T_4$ и для преодоления прямого смещения на диоде $D_1$. Транзисторы $T_4$ и $T_5$ выключены и не пропускают на выход $Y$ ток от источника +5 В. Выходные транзисторы $T_3$ и $T_4$ находятся в выключенном состоянии, и выход $Y$ не может ни отдавать ток в линию, ни принимать ток из линии. Говорят, что выход находится в высокоимпедансном состоянии, в котором он не выдает никакого логического уровня на подключенную к нему линию и почти совсем не нагружает ее.

Когда управляющий вход $C$ подключается к питанию +5 В (логическая «1»), диод $D_2$ смещается в обратном направлении и не фиксирует потенциал коллектора $T_2$. Транзистором $T_1$ теперь управляет только вход $A$: заземление $A$ вызывает включение $T_1$, а подача на вход $A$ напряжения +5 В переводит $T_1$ в выключенное состояние. Элемент ведет себя как логический инвертор — выход $Y$ всегда находится в противоположном входу $A$ логическом состоянии.

Элемент, показанный на рис. 1.6, имеет три возможных выходных состояния. Когда управляющий вход $C$ имеет уровень логической «1», выход $Y$ находится в одном из двух возможных двоичных состояний в зависимости от состояния входа $A$. Если же $C$ переводится на уровень логического «0», выход $Y$ будет в высокоимпедансном состоянии, и оно отключает элемент от линии, к которой он подсоединен. Элементы такого типа называются тристабильными элементами (драйверами) и широко применяются в вычислительных системах для подключения устройств к двунаправленным линиям шины данных.

Тристабильные элементы играют важную роль в управлении ЦП передачами по шине данных. Любая микросхема, которая может выдавать данные на шину данных, имеет один или несколько управляющих входов $\overline{CE}$ (разрешение кристалла), $\overline{CS}$ (выбора кристалла) или $\overline{OE}$ (разрешение выхода). Эти управляющие входы действуют аналогично управляющему входу $C$ в элементе, показанном на рис. 1.6, разрешая или запрещая работу драйверов шины данных в микросхеме. Например, в случае микросхемы памяти с 8 выходами ее управляющий вход либо переводит все 8 выходных ли-

ний в высокоимпедансное состояние, либо разрешает им выдать двоичные уровни на те линии, к которым они подключены. Обычно в микропроцессорной системе управляющие сигналы подчиняются стандарту негативной логики, что отмечается чертой отрицания над мнемоникой сигнала. Активным значением таких сигналов является нулевой потенциал, поэтому, например, микросхема памяти может выдать данные на шину данных, когда ее вход $\overline{CE}$ имеет состояние логического «0». В пассивном состоянии ЦП переводит все микросхемы, подключенные к шине данных, в высокоимпедансное состояние, так что ни одна из них не может выдать данные на шину. Когда же ЦП осуществляет считывание из ячейки памяти или входного порта, он селективно разрешает работу только соответствующей микросхеме, оставляя все остальные схемы запрещениями. Выбранная микросхема помещает данные на шину, с которой их считывает ЦП. После этого ЦП запрещает работу выбранной микросхемы, возвращая шину в пассивное состояние. Управление передачами информации в компьютере обеспечивается тем, что ЦП селективно разрешает выдать данные на шину данных только одной микросхеме. Это справедливо и для драйверов шины данных самого ЦП.

### 1.5. Дешифрирование адреса

В типичной вычислительной системе для хранения программ и временного хранения данных используется большое число микросхем памяти. Кроме того, имеется несколько входных и выходных портов для взаимодействия с внешними устройствами. ЦП должен селективно разрешать работу каждой микросхеме памяти, входному или выходному порту, когда они должны воспринимать данные с шины данных или помещать данные на нее. Сам ЦП не может сформировать отдельные выбирающие (селектирующие) сигналы для каждой из микросхем, имеющихся в системе. Выбирающие сигналы необходимо выделить из информации на шине адреса и сопроводить их соответствующими сигналами шины управления.

Обычно компоненты, применяемые в микрокомпьютерах, не содержат по одной ячейке памяти или отдельному входному или выходному порту. Микросхемы памяти организованы в блоки из 1К ячеек и имеют внутренний

дешифратор, обеспечивающий доступ к отдельным ячейкам. Типичная микросхема памяти содержит 1К, 2К или 4К байтных ячеек, а выбор конкретной ячейки осуществляет внутренний дешифратор по состояниям линий шины адреса. К микросхеме 1К непосредственно подключаются 10 младших линий адреса $A_0$—$A_9$, определяющих конкретную ячейку.

Конечно, имеются исключения из общего правила, когда емкость микросхемы меньше 1К или ее ячейки имеют меньше 8 бит. Общий принцип выбора микросхем остается тем же самым, а изменяются только детали подключений, которые и учитывают тип применяемых микросхем. В малых или дешевых системах применяются отдельные входные и выходные порты, что требует однозначного дешифрования их адресов по информации на шине адреса. Для упрощения схем дешифраторов портам обычно присваивается блок адресов памяти, и любой из этих адресов разрешает работу порта. Однако в настоящее время заметна тенденция использовать в микрокомпьютерах программируемые микросхемы ВВ, в которых имеются несколько ячеек, предназначенных для выполнения функций ВВ, и внутренний дешифратор для выбора конкретной ячейки.

Старшие линии адреса служат для генерирования сигналов $\overline{CE}$, выбирающих микросхемы памяти и ВВ. Рис. 1.7 иллюстрирует механизм «расщепления» шины адреса

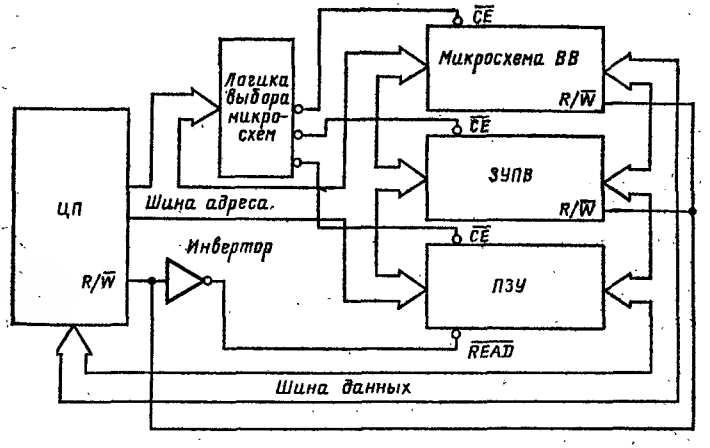



**Рис. 1.7.** Упрощенная схема дешнфрирования адреса

между микросхемами памяти и ВВ, а также логику выбора микросхем. Младшие линии адреса подключены непосредственно ко всем блокам системы, а старшие линии адреса поданы в логическую схему выбора (дешифратор). Для обращения к конкретной ячейке в системе используется весь двоичный код на шине адреса; младшие линии определяют ячейку внутри блока адресов, а старшие линии — блок адресов. При таком способе адресации микросхему можно разместить где угодно в общем пространстве памяти, а ее точное местонахождение определяется соединением входа $\overline{CE}$ с соответствующим выходом логики выбора микросхем.

В микропроцессорных системах применяются два вида полупроводниковой памяти. *Запоминающие устройства с произвольной выборкой* (ЗУПВ) выполняют операции записи и считывания, а из *постоянного запоминающего устройства* (ПЗУ) микропроцессор может только считывать информацию. ПЗУ являются энергонезависимыми, т. е. их содержимое сохраняется при отключении электропитания, а ЗУПВ энергозависимы. ПЗУ применяются для хранения программ в таких системах, где неудобно загружать программу после каждого выключения питания. К подобного рода системам относится большинство приборных и управляющих систем. Для предотвращения конфликтов на шине разрешение работы ПЗУ часто обусловлено сигналом шины управления, который идентифицирует операцию считывания. На рис. 1.7 сигнал шины управления, определяющий операции считывания или записи, показан с мнемоникой $R/\overline{W}$. Она обозначает, что операция считывания задается уровнем логической «1» на линии $R/\overline{W}$, а операция записи — уровнем логического 0. Большинство сигналов разрешения микросхем имеют активный низкий уровень, т. е. сигнал истинен при уровне логического 0, поэтому до подачи сигнала $R/\overline{W}$ в ПЗУ его необходимо инвертировать. На схемах сигналы с активным низким уровнем показываются с кружком в местах входов и выходов (см. вход $\overline{READ}$ в ПЗУ на рис. 1.7).

В микросхемах ЗУПВ управляющий сигнал $R/\overline{W}$, определяющий тип выполняемой операции, подается непосредственно на соответствующий вход. Если в линии $R/\overline{W}$ присутствует уровень логического 0, когда работа микросхемы разрешена низким уровнем сигнала

$\overline{CE}$, то выполняется операция записи, и данные с шины данных загружаются в ячейку, определяемую кодом на младших линиях шины адреса, подключенных к ЗУПВ. Когда же линия $R/\overline{W}$ имеет уровень логической 1 и работа ЗУПВ разрешена сигналом $\overline{CE}$, то выполняется операция считывания, и содержимое ячейки, выбранной младшими линиями шины адреса, помещается на шину данных.

Программируемые микросхемы ВВ в системе с отображением ВВ на память рассматриваются так же, как ЗУПВ. Считывание из входного порта осуществляется, когда он выбран, а на управляющей линии $R/\overline{W}$ действует уровень логической 1; загрузка в выходной порт производится при действии на линии $R/\overline{W}$ уровня логического 0. Когда входные и выходные порты реализуются на микросхемах с *малой* или *средней степенью интеграции,* разрешающие сигналы от логики выбора микросхем необходимо объединить с сигналом $R/\overline{W}$ при помощи внешних схем.

В небольших системах в качестве входного порта применяется микросхема 74LS244 —шинный приемопередатчик на 8 каналов. На рис. 1.8 показана схема включения 8-битного входного порта, через который микрокомпьютер может считать состояния 8 внешних сигналов. Микросхема 74LS244 имеет тристабильные выходы, поэтому, когда на ее разрешающих входах находятся уровни логической 1, выходы оказываются отключениями и не нагружают шину данных. Сигнал $\overline{CE}$ от системной логики выбора микросхем разрешает работу этой микросхемы и позволяет осуществить передачу входных сигналов на линии шины данных. Если программист ошибочно написал команду, которая попытается произвести операцию записи в такой входной

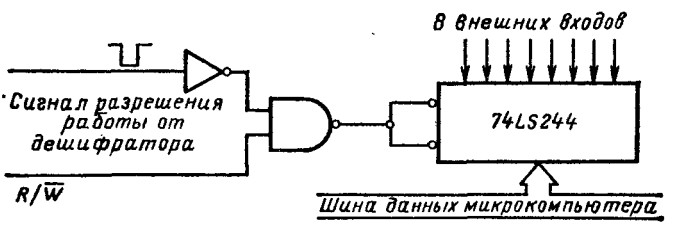



**Рис. 1.8.** Выбор простого входного порта

порт, и если не обусловить разрешение работы порта сигналом $R/\overline{W}$, то возникает конфликт на шине. Микропроцессор попытается поместить данные на шину данных, а работа входного порта будет разрешена, и порт тоже попытается выдать данные на шину данных. Эту ситуацию можно исключить с помощью схемы, показанной на рис. 1.8, где сигнал $\overline{CE}$ объединяется по *И* с сигналом $R/\overline{W}$, так что работа микросхемы 74LS244 будет разрешена только при операции считывания. В ошибочной ситуации операции записи работа входного порта не будет разрешена и конфликт на шине не возникнет.

Отметим, что на рис. 1.8 единственной адресной информацией являются старшие линии адреса, подаваемые в логику выбора микросхем. Следовательно, этот простой входной порт будет представлен в системной карте памяти всеми адресами, по которым формируется сигнал выбора блока, а однозначного адреса у него нет.

На рис. 1.9 показана схема загрузки данных с шины данных в простой выходной порт. Микросхема 74LS273 содержит 8 *D*-триггеров; их выходы *Q* подключаются к внешним устройствам, а входы *D* подсоединены к шине данных микрокомпьютера. В *D*-триггерах положительный фронт сигнала синхронизации Clock передает информацию со входов *D* на выходы *Q*. Поэтому сигнал $\overline{CE}$ инвертируется логическим элементом для передачи информации с шины данных на выходы микросхемы. Отметим отсутствие схемы, определяющей операцию «запись»; в случае выходного порта конфликт на шине возникнуть не может, так как ошибочное считывание из такого порта заставит ЦП считать пассивное состояние шины данных. Однако схему определения операции записи целесообразно ввести в систему (рис. 1.10), чтобы

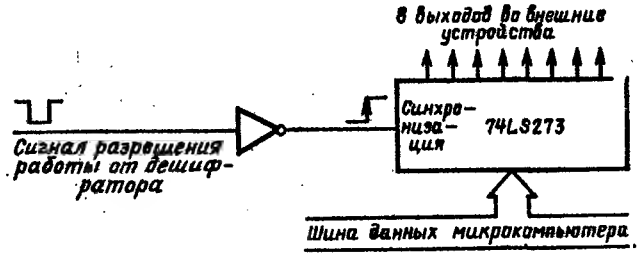



**Рис. 1.9.** Простой выходной порт

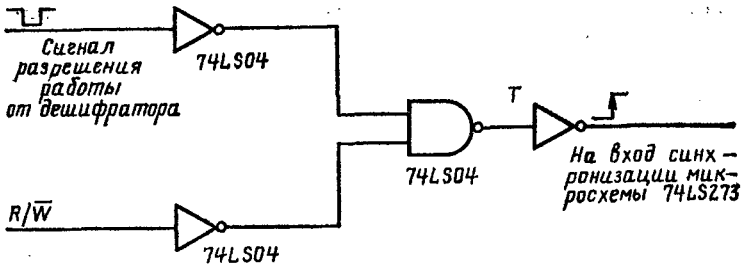



**Рис. 1.10.** Простая схема выбора выходного порта

состояния выходов микросхемы 74LS273 не изменялись из-за ошибочного задания операции считывания.

В большинстве микрокомпьютеров в качестве логики выбора микросхем применяется дешифратор. В случае 8-битного микропроцессора с 16 линиями в шине адреса все пространство памяти 64К разделяется на 16 блоков по 4К с помощью четырехвходового ТТЛ-дешифратора 74154. Рис. 1.11 иллюстрирует применение дешифратора, который декодирует 4 старшие линии шины адреса $A_{12}$—$A_{15}$ на 16 выходов по адресным границам с дискретностью 4К.

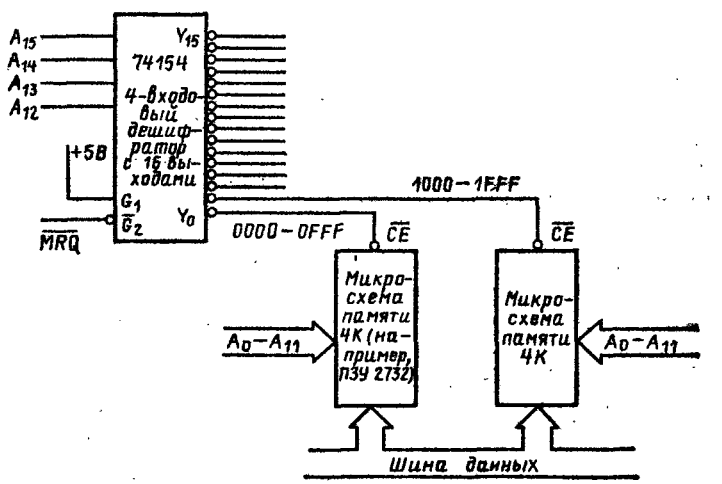



**Рис. 1.11.** Дешифрирование адреса по границам 4К

Двоичный код на четырех входных линиях вызывает на одной из 16 выходных линий изменение пассивного состояния логической 1 на активное состояние логического 0, если два разрешающих входа $G_1$ и $\overline{G}_2$ находятся в активных состояниях. Если, например, работа микросхемы разрешена сигналами $G_1$ и $\overline{G}_2$ и на входы подан двоичный код 0000, то выход $Y_0$ окажется в активном состоянии логического «0», а все остальные выходы останутся в пассивных состояниях логической «1».

На рис. 1.11 дешифратором 74154 управляет сигнал запроса памяти $\overline{MRQ}$, входящий в шину управления. Данный рисунок показывает систему с ВВ, отображенным на ВВ, так как срабатывание логики выбора микросхем формирует сигнал запроса памяти; аналогичный дешифратор необходим и для выбора портов ВВ, но с заменой сигнала запроса памяти на сигнал запроса ВВ. В системе с ВВ, отображенным на память, работа микросхем памяти и ВВ должна разрешаться от общего дешифратора адреса, который либо должен быть постоянно разрешен путем подключения его разрешающих входов на землю и к источнику электропитания, либо должен разрешаться каким-либо синхронизирующим сигналом от микропроцессора. Схему, приведенную на рис. 1.11, можно модифицировать для системы с отображением на память, подключив вход $G_1$ к питанию +5 В, а вход $\overline{G}_2$—на землю.

Во многих небольших системах, например, в специализированных промышленных контроллерах, не требуется использовать все имеющееся адресное пространство. Сигналы выбора блоков могут потребоваться на границах, меньших адресных границ 4К, показанных на рис. 1.11. Если существующие требования можно удовлетворить при наличии памяти меньше 16К и применить микросхемы с емкостью 1К, схему на рис. 1.11 следует изменить, подав на входы дешифратора 74154 линии адреса $A_{10}$—$A_{13}$. От дешифратора получается 16 выходов на границах 1К, а младшие 10 линий адреса $A_0$—$A_9$ подаются на все микросхемы памяти 1К.

В небольших системах часто применяется дешифратор 74LS138 с тремя входами и восемью выходами. Рис. 1.12 показывает адресное пространство 8К, дешифрируемое по границам 1К. Здесь с помощью линий адреса $A_{10}$, $A_{11}$ и $A_{12}$ формируются 8 сигналов разрешения бло-

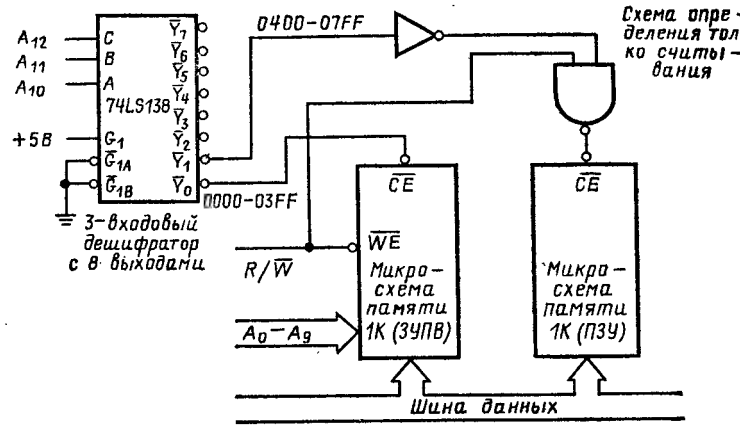



**Рис. 1.12.** Пространство памяти 8К, дешифрируемое по адресным границам 1К

ков по 1К. Отметим, что в данной схеме сигнал разрешения работы ПЗУ ($\overline{CE}$) обусловлен управляющим сигналом $R/\overline{W}$, чтобы предотвратить возникновение конфликта на шине.

При соответствующем выборе дешифраторов адреса и линий адреса можно разделить пространство памяти на блоки любого размера. Если применяемые микросхемы памяти имеют организацию, меньшую формата слова компьютера, их следует включить параллельно с образованием блока памяти, ширина которого равна требуемому формату слова.

Микросхема ЗУПВ 2114 (рис. 1.13) имеет внутреннюю организацию 1024×4, поэтому для получения памяти с байтной структурой две такие микросхемы следует включить параллельно. Младшие линии адреса и управляющие сигналы подаются в микросхемы параллельно, но шина данных «расщеплена» между ними. Младшие четыре линии шины данных $D_0$—$D_3$ подключены к одному ЗУПВ, а старшие четыре линии $D_4$—$D_7$— ко второму ЗУПВ. Когда в ЗУПВ записывается байт данных, младшая тетрада (четыре бита) записывается в $ЗУПВ_1$, а старшая тетрада — в $ЗУПВ_2$. Аналогично считывание из ЗУПВ вызывает считывание старшей тетрады из $ЗУПВ_2$, а младшей тетрады — из $ЗУПВ_1$. Восьмибитный микропроцессор выдает сразу байт дан-

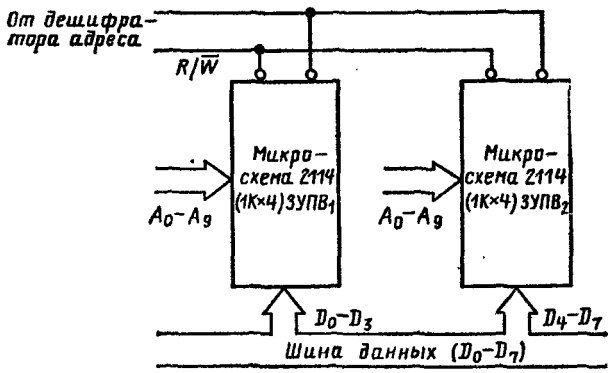



**Рис. 1.13.** Параллельное включение микросхем памяти

ных и полагает, что ЗУПВ является байтной микросхемой. Такие же рассуждения относятся и к микросхемам памяти с организацией $N\times1$, но для образования блока памяти с байтной структурой потребуется включить параллельно восемь микросхем.

### 1.6. Микроконтроллер

Микроконтроллер представляет собой устройство на основе микрокомпьютера, которое используется для решения задач управления и измерения. Программа его специализирована на выполнение ограниченного числа конкретных функций, например, на считывание сигналов преобразователей, обработку значений данных и выдачу управляющих сигналов. Требование энергонезависимой памяти предопределяет размещение программы в ПЗУ, а для хранения промежуточных данных предусматривается небольшое ЗУПВ. Микроконтроллеры отличаются от компьютеров общего назначения в основном относительным использованием ПЗУ и ЗУПВ. В машине общего назначения требуется большое ЗУПВ, так как выполняемые ею программы часто заменяются, а для этого необходимо ЗУПВ, и в то же время требуется небольшое ПЗУ для хранения часто используемых фиксированных программ. Микроконтроллер же имеет относительно большое ПЗУ для хранения своих фиксированных программ и небольшое ЗУПВ для *стека* и временного хранения данных. В микроконтроллере требуется

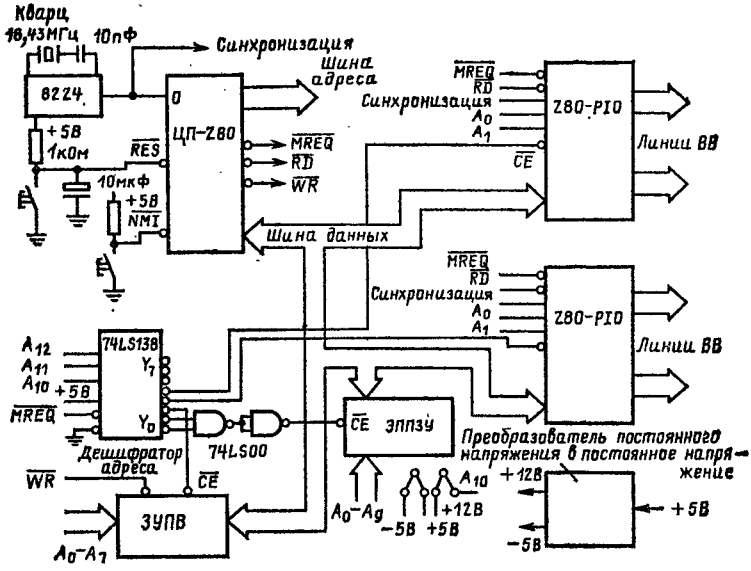



**Рис. 1.14.** Микроконтроллер на основе микропроцессора Z80

большое число линий ВВ для взаимодействия с внешними устройствами, а машина общего назначения имеет ограниченное число специальных линий ВВ, например, линий последовательной связи, для подключения видеодисплеев и принтеров.

Обычно микроконтроллеры представляют собой системы с простейшей конфигурацией и часто выполняются на одной печатной плате. Предусматривается возможность сравнительно простого расширения системы благодаря шинной структуре. При этом дополнительные микросхемы памяти и порты ВВ размещаются на других печатных платах и соединяются с микроконтроллерной платой при помощи разъемов и монтажных соединений. На рис. 1.14 показан микроконтроллер на базе микропроцессора Z80 фирмы Zilog, который может служить основой для создания разнообразных промышленных контроллеров.

К микропроцессору подключен системный генератор, реализованный на микросхеме 8224 фирмы Intel. Он формирует сигналы с частотой 2 МГц, которые синхронизируют все операции микропроцессора. Вместо простых

портов ВВ применяются программируемые интерфейсные микросхемы, что повышает гибкость применения системы. Микросхемы программируемого ввода-вывода PIO фирмы Zilog для своей работы требуют сигналы синхронизации. Обычно PIO рассчитаны на способ ВВ, отображенного на ВВ, но в данном микроконтроллере они отображены на адресное пространство памяти, при этом устраняется необходимость использования дополнительного дешифратора адреса для портов ВВ. На плате микроконтроллера имеются проводные перемычки для использования микросхем ПЗУ емкостью 1К или 2К. Изменив схему коммутации выходов дешифратора адреса 74LS138, в системе можно применять электрически программируемые ПЗУ 2732 емкостью 4К.

Небольшое ЗУПВ емкостью 256 байт реализовано на двух микросхемах 2112А статической памяти; оно предназначено для организации стека и для временного хранения вычисленных значений переменных. Чтобы получить ЗУПВ емкостью 1К, можно применить микросхемы 2114А.

Как видно из рисунка, микроконтроллер имеет ПЗУ емкостью 1К или 2К, ЗУПВ емкостью 256 байт и 32 линии ВВ. С учетом микросхемы преобразователя постоянного напряжения в постоянное напряжение, дающего питанне —12 и —5 В для микросхемы 2708, микроконтроллер реализован на 10 корпусах микросхем и может быть размещен на одной печатной плате типа Eurocard (Европлата).

При включении питания или системном сбросе микропроцессор Z80 устанавливает программный счетчик в нуль, поэтому по младшим адресам адресного пространства памяти располагается специальное электрически программируемое ПЗУ, содержащее программу инициализации системы.

### 1.7. Уровни программирования

В общем случае программирование вычислительной системы можно вести на нескольких уровнях. Для понимания причин существования нескольких уровней языков программирования и их необходимости следует рассмотреть действия ЦП при выполнении типичной команды. Каждая команда хранится в компьютере в специальном формате, который обычно имеет вид

⟨ КОД ОПЕРАЦИИ ⟩, ⟨ ОПЕРАНД ⟩, ⟨ ОПЕРАНД ⟩
Часть команды КОП (код операции) несет информацию о типе выполняемой операции, например ADD (сложить), SUBTRACT (вычесть) или MOVE (переслать), а также может содержать информацию о местонахождении участвующих в операции чисел. В приведенном типичном формате обрабатываемые числа храняться как часть команды. Обычно для перекрытия всего диапазона команд, которые может выполнять компьютер, требуется несколько форматов. Возможен, например, формат ⟨ КОП ⟩, ⟨ АДРЕС ⟩, в котором команда заставляет программу перейти к несмежной ячейке памяти, определяемой АДРЕСОМ. Каждая часть команды должна храниться в машине как совокупность двоичных кодов; машина различает коды-приказы (коды операций) и собственно числа (операнды или адреса), размещая их в памяти в определенном порядке, соответствующем форматам команд.

В большинстве компьютеров рассмотренное представление образует низший уровень, на котором оператор может программировать машину. Он называется программированием в МАШИННОМ КОДЕ и реализуется загрузкой в память соответствующих двоичных чисел, которые машина воспринимает как коды операций и значения данных, ассоциируемые с каждой командой. Для упрощения программирования обычно допускается ввод чисел в системах счисления с большими основаниями, которые тесно связаны с двоичной системой, например в восьмеричной или шестнадцатеричной. Как пример программирования на этом уровне предположим, что необходимо прибавить десятичное число 12 к внутреннему регистру ЦП, который мы назовем регистром *A*. Пусть машина воспринимает 16-ричный код *C6* как код операции. Команду следует загрузить в две смежные ячейки памяти в виде чисел *C6* и *OC*, где *OC* есть 16-ричный эквивалент десятичного числа 12. При выполнении команды к предыдущему содержимому регистра *A* будет прибавлено число 12 с образованием нового результата.

Хотя уровень машинного кода обычно является низшим уровнем, доступным программисту, компьютеры работают на еще более низком уровне программирования, называемом МИКРОКОДОМ. Каждая команда из системы команд машины, например, ранее упоминавшаяся

команда ADD, представляется в виде последовательности подопераций (микроопераций), которые передают данные из одного места системы в другое и вызывают исполнение требуемых действий. В примере с командой ADD операнд следует передать из памяти в ЦП и разместить в регистре для временного хранения; необходимо настроить *арифметико-логическое устройство* (АЛУ) на выполнение команды ADD; требуется подать два операнда в АЛУ и загрузить результат в указанный регистр. Каждая команда расчленяется на соответствующую последовательность подопераций с помощью хранимого внутри ЦП микрокода. Этот функциональный уровень программированиия определяет систему команд машины на уровне ее машинного кода и оказывается весьма сложным для использования оператором.

Выпускаются некоторые микросхемы, например секционные микропроцессоры, в которых микрокодирование (или микропрограммирование) доступно пользователю. Основными причинами такого подхода являются либо желание эмулировать существующую машину путем микропрограммирования ее системы команд, либо необходимость реализации системы команд, которой нет в существующих машинах. Являясь внутренним языком ЦП, микрокод «прозрачен» для пользователя, который оперирует командами, соответствующими макроуровню машинного кода.

В общем, когда пользователь не преследует цель создания своей оригинальной системы команд, низшим уровнем программирования для него остается машинный код. Программирование на машинном коде оказывается утомительным занятием и сопровождается большой вероятностью появления ошибок, так как каждую команду нужно вводить в машину в виде последовательности чисел, а ошибочный код операции ведет к нежелательным последствиям. Если неправильно введено значение данных, то возникает так называемый «сбой» с получением неверного результата. Сама машина не выходит из строя, но результаты оказываются совершенно неожиданными.

Чтобы преодолеть трудности программирования на машинном языке, были разработаны программы-ассемблеры, дающие пользователю возможность программировать с использованием МНЕМОНИК — эквивалентов машинных кодов операций. Вместо ввода

чисел *C6* и *OC* в команде сложения теперь можно записать ADD 12. Первым достоинством языка АССЕМБЛЕР является уменьшение ошибок из-за неправильного ввода кодов операций. Если, например, вместо ADD ошибочно задана мнемоника UDD, контролирующие процедуры в составе языка АССЕМБЛЕР обнаружат ее и сообщат об ошибке оператору. Такие мнемоники, как ADD, SUB, AND и MOV, широко применяются для указания типа выполняемой операции.

Ассемблерные программы имеют однозначное соответствие с командами машинного кода и столь же эффективны. Язык АССЕМБЛЕР обеспечивает и другие удобства программирования, например, использование в текстах программ меток для адресов и символических имен для значений переменных вместо их действительных значений. Начальный адрес программы определяется только в процессе ассемблирования, поэтому в одном прогоне программа запускается с одного адреса, но если этот же программный сегмент позднее применяется в другой программе, то его начальный адрес можно изменить. В связи с этим ассемблерная программа называется ПЕРЕМЕСТИМОЙ, так как в процессе ассемблирования, превращающего ее в эквивалентный машинный код, программу можно разместить, начиная с любого требуемого адреса.

Система команд любого компьютера ограничена по числу и типу выполняемых операций. Арифметические команды работают с целыми числами из узкого диапазона, который для типичного 8-битного микропроцессора составляет $-128 \div +127$. При необходимости обработки чисел из широкого диапазона приходится «строить» специальные программы из базовых команд, входящих в систему команд. Вычисление тригонометрических функций осуществляется путем разложения их в числовые ряды последовательностями машинных команд. При необходимости работы с иррациональными числами и сложными математическими функциями в машину встраиваются соответствующие подпрограммы с возможностью их вызова в нужных местах. Для хранения значений операндов для этих подпрограмм в памяти резервируются специальные ячейки, и после вызова подпрограмм из этих ячеек передаются операнды в специальных форматах. Обычно операнд занимает 3 или 4 байта памяти и хранится в виде МАНТИССЫ и ПОРЯДКА.

Но даже при условии такого расширения системы команда микрокомпьютера программирование на уровне языка АССЕМБЛЕР остается трудоемким занятием. Чтобы снять проблемы программирования на данном уровне, были созданы языки программирования более высокого уровня, такие, как ФОРТРАН, БЕЙСИК и ПАСКАЛЬ. В них сделана попытка сориентировать задачу программирования компьютера на людей, а не на машину. В языках ВЫСОКОГО УРОВНЯ появились инструкции (операторы), напоминающие выражения на английском языке, и такие математические символы, как значок /, обозначающий операцию деления. Большинство языков высокого уровня опирается на небольшое число формальных конструкций типа IF ... THEN ... ELSE (ЕСЛИ...ТО...ИНАЧЕ) и WHILE...DO (ДЕЛАЙ ДО ТЕХ ПОР, ПОКА...). Для каждого языка имеется набор синтаксических, определяющих формат инструкций, и в большинстве случаев требуется неукоснительно соблюдать их. Обычно в инструкциях разрешается свободно вводить пробелы, но использование символов, определяющих синтаксис инструкций, например запятых или точек с запятой, а иногда и алфавитных символов, например, первых букв имен переменных, строго регламентировано, и отступление от правил приводит к получению неработоспособной программы.

Компьютер может выполнять только команды машинного языка, поэтому программу на языке высокого уровня требуется преобразовать (транслировать) в эквивалентные последовательности команд машинного кода. Трансляция оказывается не особенно эффективной процедурой по длине получающейся программы на машинном языке, которую почти всегда можно написать более компактной и быстрой, пользуясь машинным кодом. Однако этот недостаток более чем компенсируется меньшими затратами времени на программирование с привлечением языка высокого уровня.

Имеются два основных способа трансляции — с помощью программы ИНТЕРПРЕТАТОРА или программы КОМПИЛЯТОРА. Интерпретаторы и компиляторы представляют собой совокупности программ, выполняющих операции трансляции. Принципы трансляции, на которых они базируются, различаются в нескольких важных аспектах. Интерпретатор воспринимает одну строку программы на языке высокого уровня, преобразует ее в эк-

вивалентную последовательность команд машинного кода, а затем исполняет эти команды. Закончив с одной строкой, интерпретатор переходит к следующей, транслирует и выполняет ее. В компьютере не существует машинного кода для всей законченной программы, поэтому при новом выполнении той же самой программы интерпретатор должен вновь транслировать ее строка за строкой. Интерпретаторы особенно неэффективны, когда повторяющаяся последовательность операций выполняется в нескольких местах программы. В идеальном случае такую последовательность следует вычленить из основного тела программы и оформить в виде подпрограммы, которая вызывается по мере необходимости. Вместе с тем программа-интерпретатор и интерпретируемая прикладная программа занимают малый объем памяти системы. Интерпретаторы оказываются также относительно медленными, так как каждую строку программы необходимо транслировать отдельно.

Компилятор сразу транслирует всю программу в эквивалентный машинный код, запоминает его и затем выполняет. Следовательно, компилятор нужен только для преобразования программы в машинный код, а при его выполнении не требуется. Процесс компилирования является относительно продолжительным, он занимает несколько минут для трансляции даже простых коротких программ; требует множества длинных и сложных программ, которые в памяти компьютера одновременно не могут быть размещены. Поэтому для хранения компилятора нужна внешняя память, например накопитель на гибком диске, и компилятор загружается в память компьютера частями, каждая из которых выполняет свои конкретные операции. Достоинство компилированного варианта программы заключается в том, что она существует как полная программа в машинном коде и выполняется быстрее, чем с использованием интерпретатора. Некоторые компиляторы отыскивают в программе общие блоки и оформляют их в виде подпрограмм, что помогает значительно сократить длину прикладной программы.

Имеется множество языков высокого уровня, большинство из которых ориентировано на удовлетворение запросов определенных групп пользователей. Программисты, работающие в области искусственного интеллекта, обычно пользуются языком LISP, а занятые ком-

мерческими расчетами, — языком КОБОЛ. Основные различия между языками высокого уровня заключаются в том, как они представляют, хранят и обрабатывают исходную информацию. Но все же любой язык высокого уровня представляет собой только один уровень в иерархии средств программирования. В настоящее время разрабатываются уровни программирования, которые находятся выше современных уровней и которые сделают задачу программирования относительно простой по сравнению с сегодняшними языками высокого уровня. Существующие системы пока требуют трансляции с любого уровня до машинного кода, который оказывается единственным уровнем (за исключением «прозрачного» уровня микрокода), на котором может работать компьютер.

## 2. Проблемы тестирования систем

Цифровые вычислительные системы ставят уникальные проблемы поиска неисправностей, что объясняется их организацией и структурой. Как и в других электронных системах, компоненты компьютера по разным причинам отказывают, что приводит к частичному или полному выходу систем из строя. Архитектура вычислительной системы часто превращает поиск неисправного компонента в сложную и длительную процедуру.

Значительная часть компонентов компьютера реализована на *больших интегральных схемах* (БИС), благодаря которым законченные подсистемы компьютера поставляются в виде готовых блоков. Примером такого блока в виде БИС служит сам микропроцессор. По мере все большего уплотнения элементов в БИС проблемы их функционального тестирования резко усложняются. Любая БИС до поставки подвергается лишь ограниченному тестированию, а отказы могут проявиться при эксплуатации из-за возникновения комбинаций сигналов, которые не применялись в процессе тестирования, или таких изменений некоторого параметра компонента, которые вызывают нехарактерное поведение микросхемы.

Правильная работа вычислительной системы требует исправных аппаратных средств и безошибочного про-

граммного обеспечения. Часто это не так, и программы, считавшиеся полностью работоспособными, в определенных обстоятельствах могут давать неправильные результаты. Программа может работать правильно на конкретных тест-наборах, но отказывает при эксплуатации из-за получения непредвиденного набора кода, который она не может обработать. Такие ситуации возникают в тех программах, которые вследствие плохой конструкции не допускают ввода информации неопытным оператором, что приводит к кажущемуся отказу системы.

Кроме таких видов отказов имеются и отказы, возникающие в системе после некоторого срока эксплуатации. В общем случае они вызываются электрическими отказами из-за неисправностей компонентов или, что происходит гораздо реже, программными отказами из-за искажения хранимых в памяти кодов команд.

### 2.1. Аппаратные средства или программное обеспечение?

Двойственная природа вычислительной системы при возникновении отказа сразу же ставит следующую проблему. Если отказ не очевиден, исследователь должен решить, скрыт отказ в аппаратных средствах или вызван ошибкой в программе. На этот вопрос часто ответить не так просто, так как характер отказа может препятствовать выполнению простых диагностических тестов. Отказ в шине управления микропроцессора может препятствовать выполнению любой программы, а отказ в операционной системе может не допустить загрузку и выполнение любой тест-программы.

В небольших системах (в микроконтроллерах) простое средство в виде подключаемого вручную набора тест-программ, которые работают в том же адресном пространстве, что и прикладные программы, часто вскрывает природу отказа. Если тест-программы не работают, вероятно, имеет место отказ в аппаратных средствах, а вся тест-последовательность проверит главные функциональные блоки системы и покажет, не кроется ли отказ в прикладных программах. Ввести такое тест-ПЗУ в систему можно несколькими способами. В наиболее простом варианте следует удалить системное ПЗУ и вставить в его панельку (сокет) тест-ПЗУ. При таком подходе используются те аппаратные средства системы, которые

функционируют и в рабочем режиме, например соединения между ЦП и панелькой системного ПЗУ. Еще один способ заключается в том, что нужно встроить тест-ПЗУ в систему, а при тестировании системы тумблером переключить сигнал разрешения кристалла с системного ПЗУ на тест-ПЗУ. Конечно, мы привели не единственные варианты встраивания контролирующих средств в систему, и в дальнейшем будут рассмотрены другие варианты.

Таким образом, проблема определения того, скрыт отказ в аппаратных средствах или в программном обеспечении, показывает, что изготовители микропроцессорных систем серьезно отстают в части включения в них контролирующих средств, которые могли бы значительно упростить работу обслуживающего персонала.

## 2.2. Передачи параллельных кодов последовательно во времени

В любой момент времени полное состояние микропроцессорной системы определяется уровнями сигналов на множестве линий. В 8-битном микрокомпьютере текущий адрес представлен состояниями 16 параллельных линий *шины адреса*. В то же время данные, которые считываются или записываются по этому адресу, являются двоичным кодом на 8 параллельных линиях *шины данных*. Для определения типа выполняемой операции необходимо также знать состояния нескольких линий *шины управления*. Следовательно, для получения всей информации о передаче по шине должны быть известны состояния многих линий. Любой «кусочек» информации обычно существует на шине только в течение очень короткого временного интервала. Байт считываемых из памяти данных помещается на шину данных только на один период системной синхронизации длительностью, например, 250 нс.

Информация пересылается в микрокомпьютере параллельными кодами и последовательно во времени, поэтому каждый «кусочек» информации существует в системе очень короткое время, а затем замещается другим. Обычные контрольно-измерительные приборы, например *осциллограф*, не рассчитаны на восприятие и индикацию информации, представленной в такой форме, поэтому

необходимы специальные приборы, которые могли бы фиксировать и индицировать данные в системе с шинной структурой.

### 2.3. Мультиплексирование шины

Ограничения на размеры корпусов БИС, в частности микропроцессоров, приводят к тому, что для всех сигналов на корпусе БИС не хватает внешних выводов. При этом некоторые выводы приходится использовать для нескольких функций, что приводит к необходимости разделения (мультиплексирования) сигналов во времени. Проблема нехватки выводов особенно остро стоит в 16-битных микропроцессорах, например в приборах Z8000 фирмы Zilog и 8086 фирмы Intel. Но эта проблема существует и в 8-битных микропроцессорах. Например, в микропроцессоре Z80 фирмы Zilog сигналы $\overline{M1}$ и $\overline{IORQ}$ совместно действуют как сигнал подтверждения прерывания, а в микропроцессоре 8085 фирмы Intel шина данных мультиплексируется с младшими 8 линиями шины адреса.

Мультиплексирование шины усложняет тестирование, так как в любой момент времени приходится решать, какая информация находится на мультиплексируемой линии или линиях. Вновь приходится констатировать, что обычные контрольно-измерительные приборы совершенно не приспособлены к демультиплексированию информации в таких линиях и что необходимо применять специальные приборы.

### 2.4. Проблемы тестирования микросхем

Чтобы показать проблемы тестирования БИС более наглядно, определим время, необходимое для тестирования типичной микросхемы. Для проверки такого сложного прибора, как микропроцессор, на каждую команду из его системы команд с учетом всех возможных двоичных наборов для каждой команды необходимое число тест-комбинаций определяется следующей формулой:

$$C = 2^{mn},$$

где $n$ — длина слова данных в битах, а $m$ — число команд в системе команд микропроцессора.

Рассмотрим, к примеру, микропроцессор 8080, имеющий 8-битную шину данных и примерно 76 команд: $n=8$ и $m=76$. Общее число тест-комбинаций для полной проверки микропроцессора составит

$$C = 2^{8\times 76} = 2^{608}.$$

Преобразуем это число в более удобную для нас десятичную систему счисления.

Если

$$2^{608} = 10^{x},$$

то

$$x = \log_{10}(2^{608}) = \log_{10} 2 \cdot 608 = 0{,}30103 \cdot 608 = 183{,}02624.$$

Следовательно, общее число тест-комбинаций

$$C = 10^{183{,}02624}.$$

Предположим, что каждый тест длится 1 мкс (почти недостижимое значение для микропроцессора 8080). Тогда на проведение всех тестов потребуется $10^{183{,}02624}/10^{6}$ с $= 10^{177{,}02624}$ с. В 365-дневном году содержится $3{,}1526\times 10^{7}$ с. Поэтому выполнение всех тестов закончится через $10^{177{,}02624}/3{,}1526\times 10^{7} = 0{,}3171\times \times 10^{170{,}02624}$ лет. Если начать проверку микропроцессора сейчас, она закончится через $3{,}171\times 10^{169}$ лет!

Примерный возраст Земли оценивается в $4{,}7\times 10^{9}$ лет, поэтому невероятно, чтобы микропроцессор «выжил» хотя бы ничтожную часть требуемого времени, и уж наверняка он превратится в пыль до истечения вычисленного срока.

Приведенные расчеты свидетельствуют о том, что «сложную цифровую микросхему, например микропроцессор, никогда нельзя проверить полностью».

Отсюда следует, что и каждый существующий компьютер никогда не проверялся и не может быть проверен полностью; в лучшем случае для проверки его функционирования применялось весьма ограниченное подмножество команд и двоичных наборов. Изготовители микросхем и автоматического испытательного оборудования для БИС решают эту проблему, контролируя основные функции с помощью ограниченного множества двоичных тестнаборов и полагая по результатам такого контроля, что БИС будет правильно работать на всех тест-наборах.

С увеличением плотности упаковки, например при переходе к 16-битным микропроцессорам, проблема тестирования становится еще более актуальной.

Когда микросхема спроектирована и изготовлена опытная партия, внутренняя топология микросхемы может вызвать эффект так называемой «чувствительности к набору», заключающийся в том, что определенный двоичный набор приводит к неправильной работе. Мало вероятно, чтобы такой набор появился в тестах при производстве, его действие проявляется только в условиях эксплуатации. Драйверы выходных линий обычно размещаются на периферии кристалла, а по условиям работы они рассеивают большую мощность по сравнению с другими элементами схемы. Находящиеся вблизи драйверов маломощные схемы могут изменить свои характеристики в связи с повышением температуры. Следовательно, через некоторое время после подачи питания поведение микросхемы может стать непредсказуемым.

### 2.5. Системное ядро

Для поддержания работоспособности системы не должно быть отказов в некоторых ее компонентах. Эти важнейшие компоненты в совокупности называются системным ядром, и в него обычно входят ЦП, системный генератор синхронизации, шина управления и шина адреса.

Цифровой компьютер можно рассматривать как ядро, окруженное периферийными схемами, и ядро должно работать, чтобы можно было проверить остальные компоненты системы. К счастью, системное ядро проверяется довольно просто, если в системе предусмотрены средства, которые позволяют разорвать шину данных и ввести команду в ЦП. Обычно в ЦП вводится какая-нибудь разновидность *холостой команды,* например NOP или MOV *A,A*. Шину данных можно разорвать с помощью скользящих переключателей, отсоединяющих ЦП от шины данных. Вводимая в ЦП однобайтная команда определяется положениями одного или нескольких тестовых переключателей, с помощью которых задаются состояния линий данных, идущих в ЦП. После этого системное ядро можно перевести в режим СВОБОДНОГО СЧЕТА. Восемь скользящих переключателей, показанных на рис. 2.1 и обозначенных символом $S_1$, позволя-

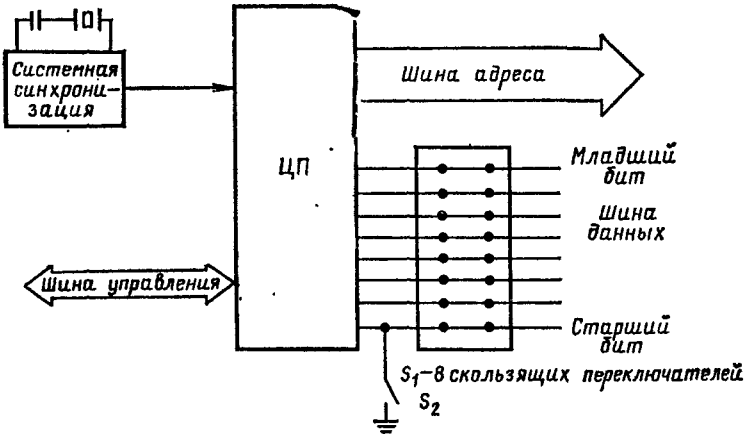



**Рис. 2.1.** Системное ядро

ют отключить ЦП от шины данных. Переключатель $S_2$ замыкает старшую линию на землю, поэтому ЦП из любой ячейки памяти считывает шестнадцатеричный код *7F*. Показанную схему применяют для реализации свободного счета в микропроцессорах 8080, 8085 и Z80.

Когда с помощью $S_1$ шина данных разорвана и $S_2$ замкнут на землю, ЦП осуществляет операцию считывания из памяти для выборки следующей команды. Она всегда интерпретируется как команда типа NOP, что заставляет ЦП перейти к следующему адресу памяти и произвести еще одну операцию считывания. ЦП вынужден считывать команду «нет операции» из каждой ячейки памяти, в результате чего на шине адреса формируются все возможные двоичные коды. При просмотре сигналов в каждой линии шины адреса можно установить факт ее отказа, проявляющийся в замыкании на землю или на питание, в обрыве линии или в замыкании на другую линию шины адреса.

Если по линиям шины адреса передаются правильные сигналы, можно считать, что системное ядро функционирует. Отказ в системном генераторе синхронизации или неисправная линия шины управления почти наверняка не дадут правильной работы в режиме свободного счета и заставят обслуживающий персонал исследовать различные части системного ядра. Тест свободного счета обеспечивает простой метод тестирования некоторых

важных компонентов микропроцессорной системы и применим к любому микропроцессору. Режим свободного счета имеет также большое значение для *сигнатурного анализа,* о котором речь пойдет в гл. 7.

### 2.6. Тестирование ЦП

Являясь одним из наиболее сложных компонентов в системе, микропроцессор оказывается и наиболее надежным. Как уже говорилось, такую сложную БИС, как микропроцессор, невозможно проверить полностью, что заставляет их изготовителей ограничиваться функциональным тестированием микросхем. Такое тестирование означает, что каждая функция проверяется с привлечением ограниченного множества двоичных наборов.

Простейший вид тестирования ЦП в условиях эксплуатации — перевод системы в режим свободного счета. Он показывает, что ЦП правильно считывает команду с шины данных, формирует адресные наборы на шине адреса и правильно реагирует на сигналы системной синхронизации. Контролируя сигналы в линиях системной шины, например в линии $R/\overline{W}$, можно частично проверить шину управления.

Обычно при проверке в условиях эксплуатации ЦП считается работоспособным, и специальных проверок для него не производится, за исключением перевода системы в режим свободного счета. Почти всегда тестирование осуществляется с помощью некоторой стимулирующей программы, контролирующей систему, а это предполагает способность ЦП выполнять тест-программы.

Отказавший ЦП не может проверить самого себя, что заставляет разработчика системы, который должен предусмотреть возможность контроля ЦП, включить в систему второй ЦП только для проверки первого ЦП. Очевидно, общая сложность системы с появлением аппаратных средств второго ЦП резко возрастает, что приводит к ухудшению ее надежности. Данный подход в «чистом» виде применяется редко из-за возникающих проблем тестирования второго ЦП и его схем. Однако идея использования одной вычислительной системы для проверки другой оказалась жизнеспособной, и большинство сложных контрольно-измерительных приборов для проверки микропроцессорных систем сами имеют встроенные микропроцессоры.

## 2.7. Тестирование ПЗУ

Термин *постоянное запоминающее устройство* (ПЗУ) относится к нескольким типам микросхем, которые отличаются способом загрузки в них программы (т. е. способом их программирования). Как предполагает само название, при обычном выполнении программы из ПЗУ можно только считывать, а записывать в них невозможно. Термин ПЗУ обычно относится к микросхемам, содержимое которых «записывается» рисунком топологии взаимосвязей в ходе технологического процесса изготовления ПЗУ. Хранимые двоичные коды зафиксированы фотошаблоном окончательной металлизации, и после герметизации корпуса БИС изменить хранимые программы невозможно.

Термин ППЗУ, или *программируемое* ПЗУ, относится к тем микросхемам, в которых запоминание двоичных кодов программы осуществляется с помощью плавких перемычек. Каждый хранимый в микросхеме бит содержится в «элементе памяти», состоящем из одного транзистора. Обычно применяется биполярный транзистор с плавкой перемычкой в цепи эмиттера. В процессе программирования перемычки либо сохраняются нетронутыми, либо расплавляются проходящим через них током около 1 А. ППЗУ программирует пользователь, а в поставляемой микросхеме имеются все плавкие перемычки. Пользователь селективно «выжигает» перемычки, формируя двоичные коды прикладной программы, и после этого изменить хранимые коды нельзя, что свойственно для ПЗУ. Строго говоря, некоторую модификацию программы можно осуществить и после программирования ППЗУ, так как сохранившиеся перемычки можно разрушить при повторной операции программирования. Примером может служить ситуация, когда какая-то константа при первом программировании была неизвестна и все соответствующие перемычки остались целыми. После определения константы осуществляется повторное программирование ППЗУ.

В микросхемах *стираемых программируемых* ПЗУ (СППЗУ) допускается стирание их содержимого, обычно при помощи ультрафиолетового излучения с фиксированной длиной волны, а электрическое программирование их осуществляет пользователь. Таким образом, благодаря стиранию и репрограммированию в СППЗУ

можно хранить и модифицировать хранимые программы.

Микросхемы *электрически изменяемых* ПЗУ (ЭИПЗУ), которые называются памятью в основном со считыванием, допускают репрограммирование при обычном использовании их в вычислительной системе. Они были разработаны для удовлетворения потребности в энергонезависимых ЗУПВ.

Тип применяемых в системе микросхем зависит от объема выпуска и назначения системы. Изготовитель микросхемы должен сделать фотошаблон для ПЗУ под двоичные коды программы пользователя, что связано с относительно высокими первоначальными расходами. После этого возможен массовый выпуск дешевых ПЗУ. Следовательно, ПЗУ целесообразно применять в массовой продукции, когда расходы на фотошаблон распределяются между тысячами изделий.

Когда объем производства ниже уровня, оправдывающего применение микросхем ПЗУ, наиболее экономичными оказываются ППЗУ.

При малом объеме выпуска и в процессе разработки СППЗУ представляют собой довольно гибкую, хотя и относительно дорогую память. Подавляющее большинство микросхем полупроводниковых ЗУПВ являются энергозависимыми, т. е. их содержимое теряется при выключении питания, а ПЗУ оказываются энергонезависимыми. В памяти компьютеров старых образцов для хранения двоичных наборов применялись ферритовые сердечники, поэтому оперативная память была энергонезависимой. Энергонезависимость является желательным свойством памяти вычислительной системы, так как без нее при выключении и восстановлении питания в систему необходимо загружать программу заново. ПЗУ энергонезависимы, а ЗУПВ являются энергозависимыми, поэтому в тех ситуациях, когда компьютер вычисляет значение некоторой переменной и хранит его для использования в последующих вычислениях, неисправность питания может иметь катастрофические последствия или, как минимум, потребует сброса системы при включении питания. В некоторых системах, например в военных системах наведения летательных аппаратов, требуется энергонезависимое ЗУПВ, поскольку в процессе полета константы модифицируются. Память на ферритовых сердечниках довольно громоздка, и ее заменяют на микросхемы ЭИПЗУ. В таких системах стоимость несуществен-

на, а определяющими становятся размеры и масса.

Подавляющее большинство микросхем ПЗУ хранят фиксированные команды, которые при обычном выполнении программы не изменяются. Фиксированный характер хранимой в ПЗУ информации позволяет организовать контроль достоверности содержимого микросхем ПЗУ. Предположим, что в микросхеме ПЗУ хранятся всего два шестнадцатеричных числа 06 и 07. Сложение этих чисел дает шестнадцатеричный результат 0D, который фиксируется в последней ячейке микросхемы ПЗУ. Во время проверки выполняется программа, которая суммирует все содержимое ПЗУ, за исключением последней ячейки, а затем сравнивает свой результат с содержимым последней ячейки. Если два числа одинаковы, содержимое ПЗУ считается правильным, а при их различии делается вывод о том, что в ПЗУ возник отказ. Это число называется контрольной суммой CHECKSUM, а рассмотренный способ обеспечивает простую и эффективную проверку микросхем ПЗУ с фиксированной хранимой информацией.

Способ CHECKSUM применим к микросхемам с любой емкостью, если суммировать все их содержимое, за исключением последней ячейки, и игнорировать возникающие переполнения. Когда, например, в ПЗУ содержатся три 16-ричных числа 7C, 20 и 8A, результатом их сложения будет двухбайтное число 126. Если каждая ячейка ПЗУ представлена байтом, то в качестве CHECKSUM будет запомнена только младшая часть результата 26. Переполнение в следующий старший байт результата должна игнорировать и тест-программа, чтобы ее результат 26 можно было сравнить с хранимым значением контрольной суммы. Многократные ошибки в ПЗУ могут дать результат, который окажется равным байту истинного результата. В этом случае тест-программа посчитает его за правильный результат, и ПЗУ пройдет как исправное, хотя в нем имеются ошибки. К счастью, вероятность такой ситуации невелика.

Когда в системе программа хранится в нескольких микросхемах ПЗУ, последней командой в первом ПЗУ должна быть команда безусловного перехода, которая обходит ячейку контрольной суммы. В каждой микросхеме последняя ячейка отводится для контрольной суммы содержимого, и для их пропуска предусматриваются команды безусловных переходов.

Обычно тест-программа контрольного суммирования встраивается в системное тест-ПЗУ, которое начинает работать сразу после включения вычислительной системы. В идеальном случае тест-ПЗУ должно проверять и само себя.

## 2.8. Тестирование ЗУПВ

Термин *запоминающее устройство с произвольной выборкой* (ЗУПВ) относится к микросхемам памяти, которые при выполнении программы осуществляют операции считывания и записи. Этот термин происходит от названия запоминающих устройств прежних компьютеров, в которых был последовательный доступ от первой ячейки до требуемой ячейки. Чтобы отличить такую память от памяти, в которой адрес подается непосредственно в микросхему для прямого доступа к нужной ячейке, был использован термин «произвольная выборка». В этом смысле подавляющее большинство запоминающих устройств обеспечивают произвольную выборку, но термин ЗУПВ закрепился за устройствами, осуществляющими операции считывания и записи.

Микросхемы полупроводниковых ЗУПВ в основном энергозависимы, т. е. при выключении питания их содержимое теряется. В отличие от ПЗУ содержимое ЗУПВ не фиксировано, поэтому простой тест CHECKSUM для них не применим. Имеются два основных типа ЗУПВ: в *статических* ЗУПВ каждый бит записан в «запоминающий элемент», который сохраняет свое состояние до записи в него противоположного логического состояния; в *динамическом* ЗУПВ запоминающий элемент имеет более простую структуру, что увеличивает емкость микросхемы, но со временем информация из него исчезает. Обычно в динамическом ЗУПВ бит хранится в виде заряда на емкости затвор — исток МОП-транзистора. Этот заряд исчезнет, если его периодически не восстанавливать в цикле регенерации. Большинство динамических ЗУПВ должно иметь период регенерации 2 мс. Для регенерации динамических ЗУПВ в системе предусматриваются дополнительные цифровые схемы. Чтобы устранить этот недостаток, были разработаны микросхемы ЗУПВ со встроенными схемами регенерации. Эти микросхемы называются квазистатическими, так как, являясь дина-

мическими ЗУПВ, они взаимодействуют с остальными компонентами системы как статические.

Для проверки микросхем ЗУПВ необходимо записать данные в каждую ячейку, а затем считать их. Если считанный из ячейки двоичный код совпадает с записанным, полагается, что ячейка исправна. Записываемые и считываемые из ЗУПВ коды называются ШАХМАТНОЙ ДОСКОЙ, так как в них логические состояния чередуются, как черные и белые клетки на шахматной доске. Обычно для проверки применяются наборы 01010101 ($55_{16}$) и 10101010 ($AA_{16}$).

Тест ШАХМАТНАЯ ДОСКА применяется для проверки ЗУПВ после включения питания и позволяет определить старший адрес ЗУПВ в системе. Многие вычислительные системы поставляются в виде базовой модели с минимальным ЗУПВ, которое можно расширять до допустимого в системе максимума путем подключения дополнительных микросхем ЗУПВ. Следовательно, емкость ЗУПВ в системе может варьироваться от минимума в базовой модели до допустимого максимума. Тест-программа записывает шахматный набор в каждую ячейку ЗУПВ, начиная с известного минимального адреса ЗУПВ, и продолжает проверку по всем ячейкам до обнаружения ячейки, из которой тест-набор не считывается. Эта ячейка в системе имеет адрес, на единицу больший максимального адреса ЗУПВ, и, следовательно, определяет верхнюю границу ЗУПВ в системе. Обычно после этого теста емкость доступного пользователю ЗУПВ индицируется для оператора на видеодисплее как адрес верхней границы. Если показываемое значение меньше ожидаемого, можно предположить наличие отказавшей микросхемы, и индицируемый адрес помогает найти эту микросхему.

Во многих случаях вместо тест-наборов 55 и AA применяются более простые тест-наборы 00 и FF. Хотя шахматные наборы и служат простым средством первоначальной проверки микросхемы ЗУПВ, они не являются исчерпывающим тестом. Когда проверяется система с подозрением на отказ ЗУПВ, то запускается более жесткий тест, который заключается в записи и последующем считывании кодов «шагающих единиц». В этом тесте в каждую ячейку ЗУПВ записываются и затем считываются такие двоичные последовательности:

0000 0000
0000 0001
0000 0010
0000 0100
0000 1000
0001 0000
0010 0000
0100 0000
1000 0000

Такой тест призван обнаружить любую чувствительность соседних линий данных к перекрестным наведенным помехам, которые проявляются в двух соседних разрядах при считывании из проверяемой ячейки состояния 1.

В микросхеме ЗУПВ линии, несущие информацию о данных и адресе, расположены очень близко друг к другу, а внутренняя топология может вызвать эффект «чувствительности к наборам». Этот эффект, в котором конкретный двоичный код вызывает наводку на другие линии адреса и данных, часто не обнаруживается шахматными кодами. К счастью для большинства пользователей, чувствительные к наборам микросхемы обычно обнаруживаются при разработке или при выборочном контроле в процессе производства.

Интерференция часто возникает между линиями шины адреса из-за короткого замыкания линий вне микросхемы ЗУПВ или из-за внутреннего отказа. Для выявления всех «относящихся к адресу» проблем рекомендуется следующий тест. Первоначально во все ячейки ЗУПВ записывается код 00, а затем только в первую ячейку ЗУПВ записывается код FF. После этого осуществляется проверка ячеек микросхемы на считывание кода 00. Если тест проходит, первая ячейка сбрасывается на 00, код FF записывается только во вторую ячейку, а все остальные ячейки вновь проверяются на считывание кода 00. Если при считывании ячейка не дает результат 00, следует предположить «относящийся к адресу» отказ либо в самой микросхеме, либо во внешних линиях шины адреса. Подозреваемые бит или биты в шине адреса находятся с помощью операции исключающего ИЛИ начального адреса ЗУПВ и адреса, по которому произошло неправильное считывание. Пусть, например, базовый адрес проверяемого ЗУПВ равен $80_{16}$, а отказ обна-

ружен в ячейке $C8_{16}$. Подозреваемые линии шины адреса находятся с помощью операции исключающего ИЛИ:

| | $A_7 \ldots A_0$ | |
|---|---|---|
| $80_{16}$ | 1000 | 0000 |
| $C8_{16}$ $\oplus$ | 1100 | 1000 |
| | 0100 | 1000 |

В данном случае подозрительными оказываются линии $A_3$ и $A_6$. После этого следует проверить и локализовать отказы на обнаруженных линиях и (или) в микросхеме ЗУПВ путем ее замены.

### 2.9. Тестирование ввода-вывода

Микросхемы ввода-вывода проверять довольно трудно, так как прямые обратные связи от них в систему отсутствуют. Можно считать, что микросхемы памяти находятся в петле обратной связи, по которой информация передается в них по шине адреса, а они реагируют возвращением соответствующей информации по шине данных. Такой петли в подавляющем большинстве микросхем ВВ нет, так как они либо просто воспринимают информацию из вычислительной системы с последующим направлением ее во внешние устройства, либо пересылают информацию в вычислительную систему по запросу ЦП.

К сожалению, компоненты ВВ компьютера не только трудно адекватно проверить, но в них наиболее вероятны отказы из-за электрических перегрузок. В данном контексте мы понимаем под компонентами все интерфейсные микросхемы и порты ВВ. Например, выходная линия может быть подключена к относительно длинному кабелю, который имеет распределенную емкость и индуктивность. Логические сигналы характеризуются быстрыми переходами при переключении из одного состояния в другое. Поэтому драйвер, работающий на линию, должен быстро заряжать емкость линии, когда выходной сигнал переходит в состояние логической 1, и быстро разряжать емкость линии, когда выходной сигнал переходит в состояние логического 0. Следовательно, драйвер линии особенно подвержен перегрузкам и относится к компонентам системы, в которых наиболее вероятен отказ.

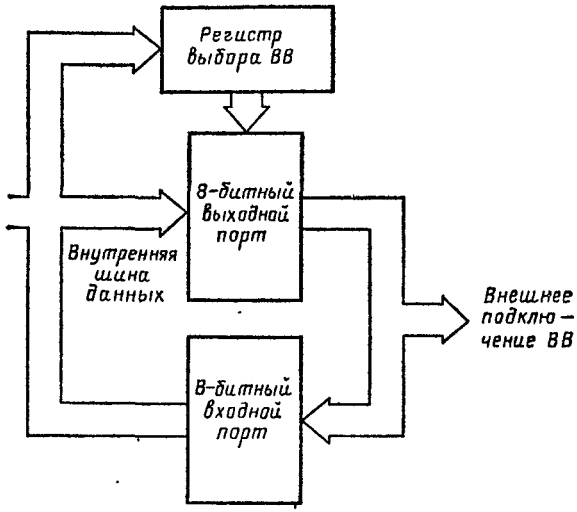



**Рис. 2.2.** Один порт программируемой микросхемы ввода-вывода

Стандартный подход к тестированию ВВ заключается в подаче известных стимулирующих сигналов через входные порты и проверке их тест-программой, а также в посылке известных сигналов из компьютера в выходные порты при наличии некоторых внешних средств индикации и контроля данных. В компьютере общего назначения, где связи портов ВВ отсоединяются, можно подключить простой тест-прибор для ввода известных тест-наборов с тумблеров, а также для восприятия данных выходного порта и отображения их на светодиодных индикаторах. Этот же подход можно реализовать и для специализированной управляющей системы, если в ней предусмотрена возможность отключения управляемого оборудования и подключения вместо него тест-прибора.

Некоторые программируемые микросхемы ВВ, например микросхема PIO фирмы Zilog, имеют внутреннюю обратную связь между выходом и входом (рис. 2.2). Когда порт запрограммирован как выходной, входной порт остается работоспособным, и, следовательно, зафиксированные в выходном порту данные можно ввести в систему, не программируя данный порт для работы на ввод информации.

Двоичный код, загружаемый в регистр выбора режима микросхемы ВВ, определяет, какие линии ВВ действуют как выходы, а какие — как входы. Если весь порт запрограммирован только как выходной, внутренние соединения между выходными и входными схемами можно использовать для считывания в вычислительную систему любых данных, зафиксированных в PIO как в выходном порту. Такая возможность позволяет проверить микросхему ВВ без подключения внешних линий ВВ. При данном способе тестирования необходимо тщательно следить за тем, чтобы в процессе выполнения тест-программы не работали никакие внешние устройства. Если микросхема входит в специализированную систему, то до начала тестирования требуется отключить все управляемое оборудование.

## 2.10. Некоторые общесистемные проблемы

Рассмотрим наиболее типичные виды отказов в вычислительных системах; механизмы некоторых отказов оказываются общими для любых электронных систем, а другие отказы специфичны только для вычислительных систем.

### 2.10.1. Блоки электропитания

По-видимому, блоки электропитания наиболее подвержены электрическим перегрузкам и отказам. В типичном компьютере для питания логических элементов применяются источники с напряжением 5 В и током более 5 А. В больших системах нагрузочная способность блоков электропитания по току достигает десятков и сотен ампер. В стандартных конструкциях для получения столь больших токов требуются громоздкие трансформаторы, мощные выпрямители и большие сглаживающие конденсаторы, а также схемы стабилизации выходного напряжения. Обычно в блок питания встраиваются схемы защиты от перегрузок по напряжению и току, чтобы предотвратить отказы в компонентах системы при выходе из строя самого блока питания. С целью уменьшения размеров низковольтных и сильноточных блоков питания их традиционные схемы вытесняются схемами блоков, работающих в режиме коммутации (импульсные блоки питания). В таких блоках выпрямленное напряжение сети переменного тока подается на высокочастотный генератор

(работающий обычно с частотой 20 кГц). Высокочастотные колебания преобразуются, выпрямляются и сглаживаются, обеспечивая низковольтное питание. Главное достоинство данного способа заключается в том, что трансформатор для высокочастотных сигналов при заданной мощности оказывается гораздо компактнее трансформатора, рассчитанного на частоту 50 Гц. В результате уменьшаются габариты всего блока питания.

Во многих системах требуются блоки питания на ±12 В для схем синхронизации, некоторых микросхем СППЗУ и схем интерфейса RS232C.

Обычно полный выход из строя блока питания можно легко обнаружить по очевидным признакам, например по выключению всех индикаторов в системе, и проверить с помощью *цифрового вольтметра.* Блок питания может казаться вышедшим из строя, если отказ в некотором блоке системы вызывает чрезмерное потребление тока. Если блок питания оборудован защитой от перегрузки по току, он может значительно повысить напряжение, но ограничить отдаваемый ток. Когда потребление тока превысит предельное значение, поведение системы станет хаотичным.

Многие блоки вычислительной системы критичны к уровням питающих напряжений и допускают лишь незначительные отклонения от их номинальных значений. Микросхема с номинальным питанием 5 В обычно допускает колебания напряжения в диапазоне 4,75—5,25 В, а вне его работает неустойчиво. Такая чувствительность к напряжению питания становится весьма критичной в тех системах, блоки которых удалены друг от друга.

Предположим, что система состоит из двух подсистем; пусть подсистема *A* находится в той же приборной стойке, в которой размещен блок питания, а подсистема *B* удалена от блока питания. Обе подсистемы подключены к блоку питания с помощью стандартного 7-жильного ленточного (плоского) кабеля, имеющего удельное сопротивление 0,06 Ом/м. Подсистема *B* расположена в 3 м от приборной стойки, в которой находятся подсистема *A* и блок питания. Для компенсации падения напряжения в кабеле в блоке питания предусмотрен «чувствительный» элемент, позволяющий устанавливать номинальное значение напряжения в некоторой удаленной точке. Выходное напряжение блока питания должно быть больше номинального значения на величину, равную падению напря-

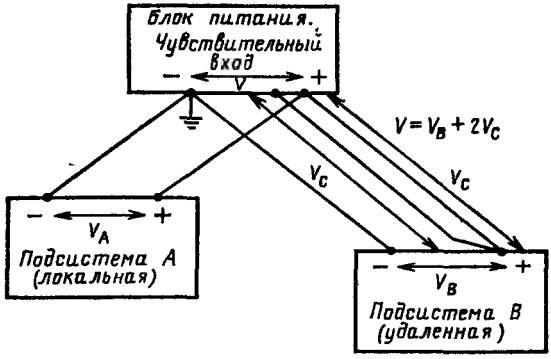



**Рис. 2.3.** Один блок питания в распределенной системе

жения в кабеле питания и в возвратном кабеле. Вход чувствительного элемента потребляет незначительный ток, и падение напряжения на соответствующем проводнике считается равным нулю.

Рассматриваемая нами ситуация представлена на рис. 2.3. Если в подсистеме *B* установлено номинальное напряжение, то напряжение на зажимах блока питания должно быть

$$V = V_b + 2V_c,$$

где $V_c$ — падение напряжения на одном из кабелей, соединяющих подсистему *B* с блоком питания. Если подсистема *B* потребляет по питанию 1 А, падение напряжения на каждом кабеле составит 0,18 В, поэтому для получения в подсистеме *B* напряжения 5 В на выходе блока питания придется установить напряжение 5,36 В. Таким образом, оно на 0,36 В выше напряжения питания в подсистеме *B*. Если предположить, что подсистема *A* расположена достаточно близко к блоку питания и сопротивлением соединительных кабелей можно пренебречь, напряжение питания в подсистеме *A* будет 5,36 В.

Изменения потребляемого подсистемой *B* тока вызовут флуктуации напряжения питания, чтобы компенсировать изменения в падении напряжения на кабеле и поддерживать $V_b$ на номинальном значении 5 В. Изменяющийся уровень напряжения подается в подсистему *A* и может вызвать ее хаотичное поведение, если напряжение питания выйдет за допустимые границы. На практике

чувствительный вход соединяется с зажимом + блока питания, поэтому блок питания сам поддерживает выходное напряжение на некотором определенном уровне. Подсистема *A* достаточно близка к блоку питания, и ее напряжение питания оказывается номинальным. Однако питание в подсистеме *B* не только будет меньше номинального значения, но и будет изменяться вместе с изменениями потребляемого тока. При таких условиях удаленная подсистема *B* будет работать неустойчиво, и оператор начнет ее проверять, при этом обычно забывают просто измерить напряжение питания.

Единственное приемлемое решение рассмотренной проблемы заключается в том, чтобы в удаленной подсистеме предусмотреть отдельный блок питания. При этом бывает достаточно ввести в имеющийся блок питания дополнительный стабилизатор.

Еще одной проблемой, связанной с блоком питания, оказываются короткие замыкания в развязывающих конденсаторах. В логических схемах на каждые два логических элемента между шинами питания включается развязывающий конденсатор емкостью 0,1 мкФ. В любой мало-мальски сложной системе появляется множество таких конденсаторов, разбросанных по схемным платам. Чтобы обеспечить в схемной плате средство хранения заряда, часто применяются малогабаритные танталовые конденсаторы значительной емкости. Обратные выбросы напряжения, наведенные в шинах питания, вызывают короткие замыкания в танталовых конденсаторах, что ведет к отключению блока питания из-за перегрузки.

На рис. 2.4 показана ситуация, возникающая при коротком замыкании одного развязывающего конденсатора. Если блок питания остается включенным (при наличии в нем схемы ограничения тока), весь ток течет через не-

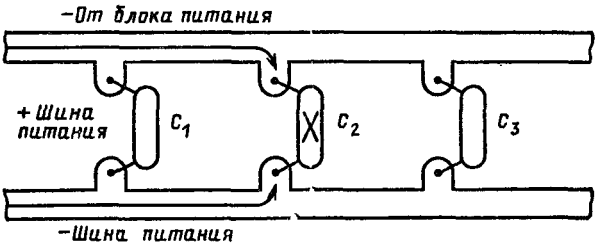



**Рис. 2.4.** Развязывающие коидеисаторы

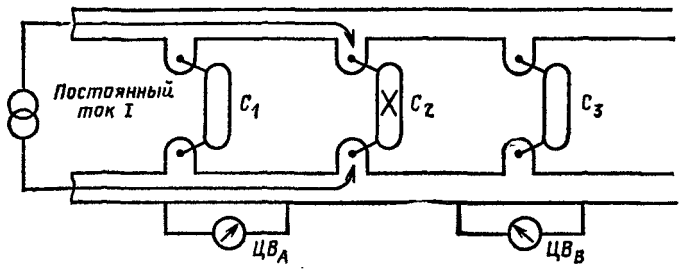



**Рис. 2.5.** Обнаружение неисправного конденсатора с помощью цифрового вольтметра

исправный конденсатор $C_2$. Все развязывающие конденсаторы включены параллельно, что затрудняет поиск отказавшего конденсатора из-за большого числа подозреваемых конденсаторов.

Обычное решение заключается в том, чтобы изолировать секции на схемной плате путем разрезания печатных шин питания и проверить каждую секцию на короткое замыкание. Ясно, что этот подход связан с повреждением схемной платы, которую после отыскания и замены отказавшего конденсатора придется ремонтировать. Как альтернативу можно предложить способ с подачей в шины питания постоянного тока и прослеживанием пути тока с помощью чувствительного *цифрового вольтметра.*

На практике воспользоваться этим способом затруднительно, так как шины питания редко проложены упорядоченно и их неудобно прослеживать. Кроме того, уровень напряжения определяется значением постоянного тока, сопротивлением медных шин и расстоянием по шине между входными контактами вольтметра. На рис. 2.5 представлен идеализированный случай, когда вольтметр *А* дает показание благодаря падению напряжения на печатном проводнике, а вольтметр *В* вообще ничего не показывает.

В гл. 5 рассматривается второй возможный способ обнаружения таких отказов, связанный с применением *индикатора тока* и обеспечивающий большую вероятность успеха.

### 2.10.2. Системная синхронизация

Генератор прямоугольных импульсов, который управляет действиями микропроцессора и всех остальных компонентов вычислительной системы, называется систем-

ным генератором синхронизации. Он участвует в каждой операции, происходящей в системе, и формирует базовый временной интервал, в течение которого может происходить событие. Отказы в генераторе либо полностью выводят систему из строя, либо приводят к бессмысленным и неопределенным программным последовательностям.

Имеется множество вариантов реализации схем генераторов синхронизации — от простых *RC*-схем до высокостабильных кварцевых генераторов. Наряду с небольшими дешевыми системами повсеместно применяются генераторы с кварцевой стабилизацией, обеспечивающие точную и стабильную частоту синхронизации.

Если системный генератор синхронизации работает слишком быстро или слишком медленно, то это отражается на устойчивости работы микропроцессорных систем. Для достижения максимальной производительности микропроцессор обычно работает на предельно допустимой частоте синхронизации, и любое превышение ее может вызвать хаотичное поведение. Большинство микропроцессоров допускает частоту синхронизации меньше максимальной, но если частота слишком мала, а в системе применяются динамические ЗУПВ, то это может привести к потере информации в ЗУПВ. Обе рассмотренные проблемы возникают с большей вероятностью, когда вместо генератора с кварцевой стабилизацией используется *RC*-генератор. Однако и кварц иногда возбуждается на третьей гармонике, формируя более высокую, чем ожидается, частоту синхронизации.

Во многих микропроцессорах требуются многофазные и неперекрывающиеся сигналы синхронизации, иногда имеющие несовместимые с ТТЛ уровни. Для микропроцессора 8080 фирмы Intel двухфазные сигналы синхронизации формируются внешней схемой, например 8224, а в более поздних микропроцессорах, например 8085 фирмы Intel, имеется внутренний генератор синхронизации. Схемы синхронизации варьируются от прямого подключения кварца или *RC*-цепочки к микропроцессору до специальной микросхемы с многофазными выходами.

К счастью, параметры сигналов синхронизации можно проверить довольно просто, пользуясь обычными *частотомерами* и *осциллографами*.

### 2.10.3. Схемы сброса

Схема сброса микропроцессора при включении питания может привести к неправильной работе. Импульс сброса, который должным образом не сформирован, имеет слишком короткую длительность, сильно забит помехами или имеет растянутый фронт, может вызвать неправильную последовательность сброса, частичный сброс или совсем не произвести сброс. В схемах сброса, подверженных помехам по цепям питания, может быть инициирован частичный сброс с неопределенным поведением микропроцессора. Выключение системы и быстрое последующее включение могут привести к такому же ее поведению, вызывая сбой в питании микропроцессора, когда его вход сброса удерживается в состоянии логической 1 конденсатором большой емкости. При этих условиях микропроцессор не сбрасывается, а его внутренние схемы находятся в неопределенных состояниях, что приводит к непредсказуемому поведению микропроцессора.

На рис. 2.6 приведена наиболее распространенная схема сброса при включении питания; при включении системы большая постоянная времени $RC$ заставляет сигнал на входе $\overline{\text{RESET}}$ находиться в состоянии логического 0 после того, как на микропроцессор подано номинальное электропитание. Такое состояние инициирует в микропроцессоре последовательность сброса при включении питания, которая обычно длится около 20 тактов синхронизации и соответствующим образом устанавливает в исходное состояние (инициализирует) внутренние регистры микропроцессора. Например, в микропроцессоре Z80 программный счетчик сбрасывается в нуль, в результате чего первую команду программы микропроцессор считывает из нулевой ячейки памяти. Продолжительность процесса сброса и осуществляемые им действия специфичны для каждого микропроцессора. Микропроцессор остается в процессе сброса до тех пор, пока конденсатор в схеме сброса не зарядится так,

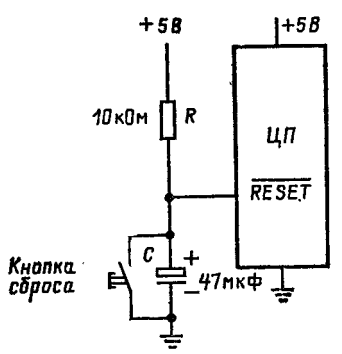



**Рис. 2.6.** Схема сброса микропроцессора при включении питания

что микропроцессор воспримет напряжение на нем как состояние логической 1.

Для сброса микропроцессора в процессе работы параллельно конденсатору включается кнопка, с помощью которой конденсатор разряжается, и на входе $\overline{\text{RESET}}$ появляется состояние логического 0.

Рассмотренная простая схема сброса может вызвать хаотичное поведение микропроцессора, если помеха приводит к моментальному отключению питания микропроцессора. При действии помехи конденсатор в схеме сброса не успевает разрядиться через резистор, сохраняя на входе $\overline{\text{RESET}}$ состояние логической 1, а состояние регистров микропроцессора искажается. Когда питание восстанавливается, микропроцессор продолжает работу с каких-то произвольных состояний внутренних регистров.

Наиболее вероятной причиной помехи в питании, достаточной, чтобы вызвать такое поведение, является моментальная неисправность в сети электропитания. Она часто называется «коричневым отключением» в отличие от более продолжительной неисправности, называемой «черным отключением». Кратковременные помехи являются случайными и непредсказуемыми событиями, поэтому в больших вычислительных системах требуются схемы обнаружения отказа сети, чтобы система могла упорядоченно отреагировать на отказ. Для удовлетворения этого требования применяются блоки питания с конденсаторами очень большой емкости, поддерживающими номинал электропитания в течение некоторого времени после отказа в сети. Кроме того, в системе предусматривается резервное аккумуляторное питание. Очевидно, в системах, где потеря информации недопустима, необходимо предусмотреть эти меры, чтобы до полного отключения система успела запомнить и сохранить информацию упорядоченным образом.

Схемы обнаружения неисправности сети обычно подсчитывают частоту сети, чтобы зафиксировать пропадание нескольких циклов и затем сформировать управляющий сигнал, инициирующий упорядоченное выключенне вычислительной системы. В небольших системах дополнительные схемы обнаружения неисправности сети обычно не оправданы, но их можно заменить более простыми схемами, которые контролируют напряжение питания.

При первоначальном включении питания (рис. 2.7) на

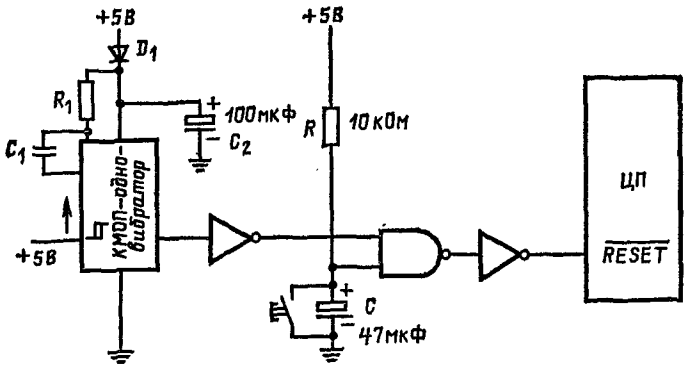



**Рис. 2.7.** Схема обнаружения неисправности питания

входе $\overline{\text{RESET}}$ поддерживается низкий уровень с помощью обычной $RC$-цепочки сброса. Если возникает помеха, которая переводит питание на низкий уровень и затем восстанавливает его, запускается КМОП-одновибратор, формирующий сигнал сброса в микропроцессор и инициирующий последовательность сброса. Одновибратор запускается нарастающим фронтом, соответствующим восстановлению питания, и имеет на входе триггер Шмитта, способный срабатывать от растянутого фронта изменяющегося уровня напряжения в цепи питания. КМОП-одновибратор на время действия помехи остается с питанием благодаря диоду $D_1$ и заряженному конденсатору $C_2$. Схема на рис. 2.7 применяется для подавления относительно кратковременных помех. Эту схему можно также использовать и для борьбы с более длительными помехами, если параллельно конденсатору $C_2$ подключить заряжаемый никель-кадмиевый аккумулятор.

Схемы сброса при включении питания можно проверить с помощью запоминающего осциллографа, фиксирующего их реакцию, или управляя их работой вручную. Постоянный низкий уровень на входе $\overline{\text{RESET}}$ микропроцессора фиксирует его в состоянии сброса и создает впечатление полностью нефункционирующей системы. Подобный отказ может возникнуть из-за замыкания на землю печатного проводника, идущего на вход $\overline{\text{RESET}}$, из-за неисправной микросхемы, подключенной к этой же линии, или из-за короткого замыкания в конденсаторе.

### 2.10.4. Прерывания

Линии прерывания, подверженные помехам или «зависшие» в одном состоянии, могут вызвать в системе неисправность. «Зависшая» линия прерывания заставляет микропроцессор обслуживать несуществующий запрос прерывания, а помеха инициирует обслуживание прерывания, которого фактически не было, что приводит к выполнению бессмысленных операций ВВ.

Структуры прерываний в микропроцессорах варьируются от простых схем с фиксированными входами, которые опрашиваются микропроцессором для определения источника прерывания, до сложных схем с программируемыми входами, реализующих механизм векторных прерываний. Число входов прерываний в 8-битных микропроцессорах изменяется от одного (вход $\overline{\text{INT}}$) до пяти, как, например, в микропроцессоре 8085 фирмы Intel. Большинство микропроцессоров имеют два входа прерываний INT и $\overline{\text{NMI}}$, причем вход *немаскируемого прерывания* $\overline{\text{NMI}}$ имеет больший приоритет.

Большинство входов прерываний в микропроцессорах воспринимают задний фронт (спад) сигнала, и изменение состояния из логической 1 в состояние логического 0 вызывает установку внутреннего триггера. Когда микропроцессор завершает выполнение текущей команды, он проверяет состояние этого триггера и, если он установлен, инициирует последовательность прерывания. Однако некоторые входы прерываний, например вход RST6.5 в микропроцессоре 8085, реагируют не на фронт сигнала, а на уровень, и поэтому при отказе микропроцессор блокируется в процедуре обслуживания прерывания. Если, например, вход RST6.5 закорочен на шину питания, микропроцессор передает управление ячейке памяти $0034_{16}$ и непрерывно выполняет начинающуюся в этой ячейке процедуру обслуживания прерывания.

Программируемые микросхемы ВВ, которые могут вызывать прерывания в системе, обычно имеют внутренние триггеры-защелки прерываний, устанавливаемые и сбрасываемые отдельно от триггера прерывания в микропроцессоре. Триггеры прерывания микросхем ВВ устанавливаются внешним сигналом запроса прерывания, а сбрасываются каким-либо сигналом подтверждения прерывания от микропроцессора, когда он начинает обслуживать запрос. Во многих микропроцессорах не хватает

выводов для всех требующихся в системе управляющих сигналов, что заставляет применять мультиплексируемые сигналы. Примером служит микропроцессор Z80, в котором сигнал подтверждения прерывания формируется с помощью двух управляющих линий, обычно выполняющих другие функции. Низкие уровни на линиях $\overline{M1}$ и $\overline{IORQ}$ образуют сигнал подтверждения прерывания; такой ситуации при обычном выполнении программы не возникает.

Программируемая микросхема PIO фирмы Zilog сбрасывает свой внутренний триггер прерывания в конце процедуры обслуживания прерывания, непосредственно дешифрируя с шины данных команду *возврата из прерывания* RETI. Эта команда содержит два байта кода операции $ED_{16}$ и $4D_{16}$, оба из которых загружаются с шины данных, когда их считывает микропроцессор Z80 в конце процедуры обслуживания. В системах, где к шине данных подключено множество микросхем ВВ, для увеличения нагрузочной способности шины применяется буферирование. Двунаправленные буферы могут воспрепятствовать PIO загрузить код команды RETI, если буферы остаются в состоянии передачи данных от микросхемы ВВ при считывании кодов RETI из памяти. В этой ситуации PIO не может воспринять коды команды RETI и не сбросит свой внутренний триггер, что заблокирует восприятие следующего сигнала запроса прерывания от внешнего источника.

Микросхема PIO представляет пользователю возможность превратить ее линии обычного ВВ во входы запросов прерываний; такой возможности нет в большинстве программируемых микросхем ВВ. В PIO любая входная линия может работать как вход прерывания. Когда таким образом используются несколько линий, то сигнал прерывания в микропроцессоре передается либо когда активна любая из линий, либо когда активны все линии. Подобная гибкость достигается в микросхеме ВВ с помощью схемы И/ИЛИ, наличие которой при определенных условиях может привести к затруднению поиска неисправности. Предположим, что схема ИЛИ запрограммирована на четыре входа запроса прерываний, и поэтому, когда любой из этих входов находится в состоянии логической 1, формируется сигнал прерывания в микропроцессор. Рассматриваемая ситуация представлена на рис. 2.8.

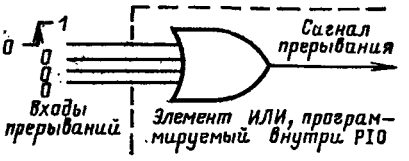



**Рис. 2.8.** Объединение по ИЛИ входов запросов прерываний в микросхеме P10

Если один из входов становится активным и переводится в состояние логической 1, то на выходе схемы ИЛИ также появляется уровень логической 1. Изменение сигнала на выходе схемы ИЛИ приводит к установке в PIO внутреннего триггера прерывания, и тем самым в микропроцессор передается сообщение об ожидающем прерывании. Микропроцессор реагирует на фронт сигнала прерывания, поэтому, если активный уровень прерывания сохраняется и после процедуры обслуживания прерывания, новые сигналы прерываний не генерируются. К сожалению, если первый сигнал прерывания «зависнет» на активном уровне логической 1, то из-за наличия в микросхеме ВВ запрограммированной логики ИЛИ никакие последующие сигналы запросов не смогут сформировать фронт сигнала прерывания в микропроцессор. Для предотвращения такой потенциально опасной ситуации процедура прерывания должна сбрасывать активное состояние линии запроса прерывания на пассивный уровень логического 0, позволяя системе реагировать на все последующие сигналы прерываний.

#### 2.10.5. Микросхемы памяти

Неисправности в памяти вызывают хаотическое поведение микропроцессорной системы разнообразными способами. Отказ может вывести из строя всю микросхему памяти или вызвать неправильное функционирование одного бита. Последствия отказа зависят от типа микросхемы и ее местоположения в карте адресов памяти.

Отказ в микросхеме ПЗУ, в общем, приводит к катастрофическим последствиям, так как в ПЗУ обычно хранятся фиксированные системные программы, которые должны функционировать правильно. Отказавший бит может находиться в неиспользуемой части микросхемы и не даст видимого эффекта, хотя наиболее вероятный результат заключается в том, что микропроцессор считает искаженное слово как ошибочный код операции и попы-

тается выполнить какую-то бессмысленную программную последовательность.

Отказы в СППЗУ могут возникать из-за частого перепрограммирования микросхемы. При программировании СППЗУ может произойти пробой каналов МОП-транзисторов, что после некоторого числа операций репрограммирования вызывает выход микросхемы из строя. В некоторых СППЗУ, например в микросхемах 1702А и 2708, такая ситуация обнаруживается по нереагирующим выводам.

ППЗУ с плавкими перемычками отказывают по причине «зарастания». В первых ППЗУ применялись нихромовые перемычки, которые после расплавления должны оставаться разомкнутыми, но спустя некоторое время они восстанавливали электрическую проводимость. Разумеется, при «зарастании» значения хранимых бит изменяются. В последующих микросхемах ППЗУ нихромовые перемычки были заменены поликремниевыми, в которых при программировании образуется слой непроводящей двуокиси кремния; эффект «зарастания» в них отсутствует.

Однобитные отказы в ЗУПВ, предназначенном для хранения данных, обычно не являются катастрофическими, но приводят к так называемым «сбоям». Сбой представляет собой ошибку из-за использования в вычислениях ошибочного значения данных. Результат окажется неожиданным, но из-за него система вряд ли выйдет из строя. Если не применять для проверки ЗУПВ при включении питания тест-программу из ПЗУ, то такие ошибки обнаружить затруднительно.

Отказ в ЗУПВ, выполняющем функции системного стека, оказывается катастрофическим, так как при возврате из подпрограммы или после выполнения процедуры обслуживания прерывания микропроцессор перейдет к неопределенной ячейке памяти.

При диагностике отказов в ЗУПВ необходимо учитывать возможность отказов в схемах регенерации динамических ЗУПВ.

#### 2.10.6. Затухание сигналов

В микропроцессорной системе линии шин адреса, данных и управления подключены ко множеству микросхем. Когда микропроцессор формирует на линии сигнал, он

распространяется по линии от микропроцессора ко всем подключенным микросхемам. Каждая подключенная к линии микросхема оказывается неоднородностью в полном характеристическом сопротивлении линии и вызывает отражение сигнала. В течение короткого временного интервала распространяющийся в линии сигнал подвергается многократным отражениям и генерации из-за несогласованных подключений к линии. Для предотвращения таких явлений в большинстве микропроцессоров отводится минимум один такт синхронизации для стабилизации новых состояний линий при выдаче на них информации. Близко расположенные на схемной плате линии подвержены перекрестным наводкам из-за многократных отражений, которые оказываются критичными для соседних сигнальных линий, несущих сигналы синхронизации и сигналы разрешения работы микросхем. Длинные, разбросанные сигнальные тракты затрудняют достижение предельных для системы временных характеристик и уровней помех, а повышенная влажность среды усиливает наводки между соседними линиями. Для проверки отдельной схемной платы в многоплатной системе плата подключается через удлинитель, что обеспечивает инструментальный доступ к находящимся на ней компонентам. Во влажной среде удлинитель сам может вызвать перекрестные наводки сигналов, если критичные к наводкам сигналы передаются по близко расположенным линиям.

Индикация почти любой сигнальной линии в микропроцессорной системе на осциллографе показывает форму сигналов, которая далека от идеализированных временны́х диаграмм, приводимых в спецификациях. Вычислительная система по принципу действия является дискретной системой, и возникающие в ней искажения сигналов компенсируются восприятием сигналов в линиях только в определенные моменты, когда все отражения в линии прекращаются. Для достижения согласования в линиях передач на концах линий помещаются согласующие резисторы, хотя на практике в большинстве систем их не применяют.

## 3. Принципы тестирования систем

Несмотря на наличие испытанных методов систематического контроля системы в случае отказа, наиболее очевидным и часто игнорируемым первым действием еще до подключения какого-либо контрольно-измерительного прибора должен быть тщательный осмотр системы. Отказ может быть очевидным, например «сгоревшая» микросхема, а состояние, в котором остановилась система, само может дать подсказку о месте возникновения отказа в системе. Необходимо проверить все встроенные в систему индикаторы, которые могут помочь в локализации отказа; кроме того, многие модульные системы имеют в подсистемах индикаторы типа «проходит/не проходит», которые идентифицируют неисправную подсистему. К сожалению, такие средства внедряются медленно, а они могут значительно сократить простой системы и свести первый этап ремонта к просмотру функциональных индикаторов, нахождению выключенного индикатора и замене неисправного модуля. Такой подход сокращает простой системы и упрощает второй этап ремонта — модуль (плата) либо отсылается на предприятие, либо ремонтируется пользователем.

Принципы поиска неисправностей в изделиях с микропроцессорами в основном не отличаются от принципов тестирования обычных цифровых устройств. В соответствии со стандартной процедурой персонал должен изучить проверяемую систему, определить состав требуемого контрольно-измерительного оборудования и составить логическую процедуру поиска неисправности. Самым сложным и совершенным прибором, находящимся в распоряжении человека, является его ум; именно на него следует полагаться на каждом этапе работ, а не подчиняться какому-то слепому набору правил.

Персонал должен изучить проверяемую систему и иметь в своем распоряжении ВСЮ требующуюся документацию. В нее входят схемы, все бланки изменений, относящиеся к конкретному варианту проверяемой системы, и, при необходимости, краткие описания схем. Нет ничего более удручающего, чем локализовать отказ в какой-то конкретной секции системы, а затем обнаружить, что схем этой секции нет или что имеющиеся схемы не соответствуют проверяемой системе, так как в них не отражены произведенные изменения. В связи с быстрым

распространением микроэлектроники персонал сталкивается со все увеличивающимся числом систем, которые сильно разняться в деталях, но которые нужно знать, чтобы грамотно выполнять свою работу. Конца этого процесса обучения не видно. Настоящая книга и аналогичные ей книги дают информацию о наиболее вероятных местах отказов и о выборе подходящей контрольно-измерительной аппаратуры, но в конце концов испытатель должен работать с одной конкретной системой, обычно имея ограниченный набор аппаратуры и часто находясь в далеко не идеальных условиях. Следовательно, в каждом конкретном случае испытатель должен опираться на свои знания системы и вероятных мест отказов и привлекать свой опыт для локализации и устранения отказа.

Диапазон контрольно-измерительной аппаратуры для микропроцессорных систем варьируется от обычных приборов, например осциллографов и цифровых вольтметров, которые применяются в данной области ограниченно, до специальных приборов, например *логических анализаторов и эмуляторов,* разработанных для решения специфических для вычислительных систем проблем. Приобретение такой аппаратуры связано со значительными расходами, и, по всей вероятности, в отделе технического контроля имеется только ограниченный перечень приборов.

Исследователь должен знать аппаратуру, представлять ее ограничения и области применения, чтобы минимизировать время поиска неисправности.

До сих пор мы касались аппаратных средств системы, но персонал должен знать и программное обеспечение, по крайней мере, те программы, которые применяются для проверки системы. Тест-программы либо встраиваются в систему и выполняются как часть процедуры включения системы, либо вызываются по требованию из тест-ПЗУ. Если ничего подобного нет, исследователь должен загрузить и выполнить свои тест-программы с какого-нибудь внешнего носителя, например кассеты, гибкого диска или подключаемого ПЗУ. Конечно, непременным условием успешной загрузки и выполнения тест-программ оказывается хорошее знание работы проверяемой системы.

Имеется несколько проблем тестирования, которые уникальны для вычислительных систем. Здесь основное управление скрыто в программах, а это затрудняет прослеживание сигналов. Еще одна проблема заключается

в том, что все процессы в системе происходят очень быстро, делая невозможным наблюдение в реальном времени, и в большинстве случаев микропроцессорную систему в отличие от обычных логических систем нельзя остановить и проанализировать. Измерения нужно производить, когда микропроцессор работает с номинальным быстродействием, и практическая необходимость фиксировать динамические события снижает эффективность многих традиционных систем контроля.

Шинная структура микропроцессорных систем ставит дополнительные проблемы, так как информация на шинах часто неустойчива и бессмысленна из-за тристабильных выходных каскадов, мультиплексирования информации на линиях и наведенных помех. Как уже отмечалось, эти особенности не вызывают проблем для самой системы, поскольку она синхронна и стробируется сигналами, совпадающими только с моментами стабильных сигналов данных. Шинная структура допускает подключение к одной линии множества микросхем, превращая проблему нахождения одной неисправной микросхемы в сложное и продолжительное занятие.

### 3.1. Программы самоконтроля

Увеличение сложности систем потребовало включить в них программы самоконтроля или диагностики. Их можно либо поместить в начальной области системного ПЗУ, находящегося в адресном пространстве памяти, там, куда переходит программный счетчик при включении питания, либо встроить в схемную плату как отдельное ПЗУ, которое подключается вместо первого системного ПЗУ и переход к которому осуществляется при сбросе вручную (рис. 3.1).

Основной недостаток реализации тест-ПЗУ, которое подключается в схему вместо обычного рабочего ПЗУ, заключаются в том, что тест-ПЗУ, конечно же, не сможет проверить заменяемое им системное ПЗУ. Если известно, что в системе имеется отказ, а все тесты проходят успешно, наиболее вероятным подозреваемым компонентом становится системное ПЗУ, которое заменено диагностическим тест-ПЗУ. После этого системное ПЗУ необходимо проверить путем замены микросхемы.

В тест-ПЗУ хранятся тест контрольной суммы ПЗУ, тест шахматной доски ЗУПВ и тесты ВВ для конкретной

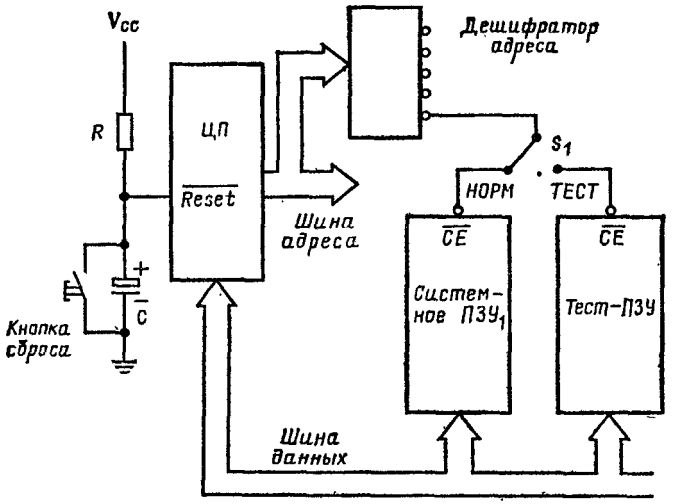



**Рис. 3.1.** Системное тест-ПЗУ

системы. Различные тест-программы можно инициировать по состояниям входов тест-порта. Такой тест-порт может быть одним из рабочих портов системы, от которого отсоединяется внешнее устройство, а вместо него подключается пульт с набором переключателей. Тесты ВВ обычно представляют собой простые программы, осуществляющие, например, посылку в выходной порт набора известных прямоугольных сигналов или выдачу сигналов двоичного счетчика, если к тест-порту подключены индикаторы. Тест-ПЗУ может содержать и тест-программы сигнатурного анализа, который рассматривается в гл. 7.

### 3.2. Существует ли отказ в действительности?

Сложность систем увеличивается по мере появления новых микросхем и применения в системах более совершенного и мощного программного обеспечения. В сложных системах пользователь зачастую может вызвать кажущуюся ошибку из-за отсутствия опыта работы, нажимая, например, кнопку, восприятие которой в программном обеспечении не было предусмотрено. Чтобы успешно пользоваться системными средствами, оператор

микрокомпьютера общего назначения должен знать не только применяемый язык программирования, но и имеющуюся операционную систему. Ошибки пользователя часто объясняются незнанием формата и синтаксиса вызовов различных программ операционной системы. Ошибка в попытке вызова программы операционной системы часто приводит к индикации сообщения об ошибке, принимаемого неопытным пользователем за отказ.

Когда подозревается отказ, часто много информации о системе можно получить, не снимая крышек оборудования. В большинстве систем имеются какие-либо клавиши для ввода и индикаторы, которые можно использовать для ввода данных и наблюдения реакций на индикаторах. Если, например, все индикаторы погашены, следует подозревать неисправность тумблера включения сети, обрыв в шнуре питания или неисправность в блоке питания. Если же не работает один из сегментов в 7-сегментном индикаторе, отказ, вероятно, возник в самом индикаторе или в схеме формирователя. Наблюдение даже за простейшими индикаторами может стать ключом к определению наиболее вероятного места отказа. Исследователь должен всегда пользоваться всеми встроенными тест-программами, например тестами при включении питания или диагностическими программами, которые поставляются с системой и инициируются внутренним переключателем.

Очень вероятно, что «тщательный осмотр» не покажет причину отказа, и исследователю придется глубже разбираться в системе. К сожалению, многие в этой ситуации следуют правилу: «если что-то неисправно, надо посмотреть описание системы». Такой, в принципе, плохой, но широко распространенный подход для микропроцессорных систем имеет еще меньше смысла, чем для обычных цифровых систем. В описании обычно есть целый набор тестовых процедур, специальные переключатели, перемычки и индикаторы, которые только и ждут, чтобы ими воспользовались. Персонал должен познакомиться со всеми этими средствами до возникновения в системе каких-либо отказов и опробовать тест-программы на исправной системе, чтобы знать возможные реакции. Все микросхемы похожи друг на друга, и исследователь должен узнать из описания системы по руководству пользователя, где находятся такие важнейшие компоненты, как микропроцессор, ЗУПВ, ПЗУ, ВВ, дешифраторы адреса, генератор синхронизации и контроллер прерываний.

Целесообразно прежде всего обратиться к тем компонентам, которые можно проверить и заменить относительно легко. Простые компоненты отказывают с такой же вероятностью, как и сложные; примером может служить блок питания. Это самый ненадежный элемент любого изделия и в то же время наиболее просто проверяемый. Выход напряжения за допустимые пределы может вызвать хаотичное поведение системы, поэтому проверка прежде всего уровней напряжения сэкономит много времени.

Полезно произвести и внешний осмотр. Грязные печатные платы и кабельные соединения вызывают в системе чрезмерный шум (обычно низкочастотный). Эти дефекты обычно можно обнаружить внешним осмотром или прикосновением. Целесообразно с известной осторожностью осуществить проверку, называемую странным термином «калиброванный кулак», при которой производится легкое постукивание по механическим узлам системы, чтобы проявились упомянутые выше дефекты. Механическое воздействие на схемную плату путем ее перекручивания и изгибания часто помогает обнаружить плохой контакт в краевом разъеме, неплотно вставленные в панельки микросхемы и разорванные печатные проводники. Большинство таких отказов быстро устраняется; например, краевые разъемы очищаются стирающей резинкой. Конечно, пользоваться описанными приемами нужно осторожно, иначе они сами могут создать в системе отказ.

Рекомендуется вести журнал проверок системы, в котором указываются характер отказа и его причина. В случае повторения отказа по журналу быстро устанавливается его причина. Ведение журнала особенно полезно, когда эксплуатируются несколько аналогичных систем, в которых обычно возникают одни и те же отказы.

### 3.3. Срок службы интегральных схем

В любой микропроцессорной системе имеется множество *интегральных схем* (ИС) — от простых элементов до таких сложных больших интегральных схем (БИС), как микропроцессор. В общем, отказы интегральных схем возникают по нескольким известным причинам и локализуются в тех секциях, которые подключены к «внешнему миру»; именно они больше всего подвержены

электрическим нагрузкам и рассеивают наибольшую мощность.

Интегральная схема производится на предприятии и приобретается изготовителем изделия, который вставляет ее в печатную плату, а сама плата устанавливается в изделие. Изделие поставляется заказчику и остается у него на весь срок службы. Нечего и говорить, что срок службы некоторых ИС оказывается довольно коротким.

Без специальной оговорки ИС поставляются с предприятия только с выборочным контролем, т. е. с производственной линии для контроля берутся отдельные образцы. Если процент брака не превышает допустимого, поставляется вся партия.

Примерно 2 % ИС, поставляемых заказчикам, оказываются дефектными. Входной контроль ИС на специальном тестере оценивается примерно в 4 пенса за штуку, но, когда ИС установлена в печатную плату, расходы на поиск отказа и ремонт возрастают примерно до 40 пенсов за штуку. Когда же дефектная микросхема на печатной плате попала в конечное изделие, приведенные расходы на поиск неисправности и ремонт подскакивают до 2 фунтов стерлингов. Замена дефектной ИС в условиях эксплуатации обходится еще дороже — примерно 40 фунтов стерлингов.

В общем, расходы на поиск неисправной ИС увеличиваются примерно в 10 раз на каждом этапе от ее приобретения до эксплуатации изделия. Очевидно, экономические соображения настоятельно требуют выявления дефектных компонентов как можно раньше. Многие изготовители систем не производят входной контроль компонентов и полагаются на долгосрочные испытания изделия. Они осуществляются на уровне печатных плат и заключаются в длительной работе оборудования, часто при повышенной температуре для ускорения проявления отказов компонентов.

Кривая интенсивности отказов ИС напоминает «поперечное сечение ванны». Сразу после выпуска ИС подвержены отказам по многим причинам, среди которых укажем плохую герметизацию корпуса, ненадежные внешние соединения, чувствительность к кодовым комбинациям при повышенной температуре и др. Период приработки в 100 ч обычно вызывает отказы в критичных микросхемах, которые могли пройти все первоначальные проверки. Если ИС «выживает» первые 100 ч

своего срока службы, в течение последующих восьми лет вероятность ее отказа мала. Постоянные электрические нагрузки в течение срока службы увеличивают вероятность отказа, что объясняет «загибание» вверх кривой интенсивности отказов. Перегрузка, даже кратковременная, ускоряет процесс «старения»; примером служат отказы из-за частого репрограммирования СППЗУ.

Интегральная схема представляет собой корпус, в котором размещен кристалл кремния. Соединения между выводами корпуса и контактными площадками на кристалле выполняются золотыми проводниками с помощью ультразвуковой сварки. Наиболее вероятный отказ в ИС — это разрыв проводника, вызванный либо механической вибрацией, либо чрезмерным током. Воздействие такого отказа на схему, в которой находится неисправная ИС, зависит от того, является отказавшее соединение входным или выходным. Отказавшее входное соединение блокирует распространение сигнала, и его можно обнаружить, только отмечая отсутствие влияния изменения входного сигнала на какой-то выход микросхемы. Отказавшее выходное соединение в случае ТТЛ-схем проявляет себя как неопределенный логический уровень с напряжением около 1,5 В.

Еще три вида отказов возникают внутри самой ИС. Входной или выходной вывод может быть закорочен на питание $V_{cc}$ или на землю; появляется короткое замыкание между двумя выводами, но без замыкания на $V_{cc}$ или на землю, или возникает отказ во внутренних элементах ИС.

Отказ наиболее вероятен при первом включении изделия на предприятии, когда отказать может что угодно. Типичными отказами, появляющимися после сборки изделия, являются «выбросы» припоя, закорачивающие печатные проводники, неправильные размещения микросхем и плохие контакты. Изделия, отказавшие при эксплуатации, когда-то работали некоторое время; отказы, наиболее распространенные в производстве, обычно можно устранить с помощью простых принадлежностей.

### 3.4. Тестирование нагрузками

Тестирование нагрузками заключается в том, чтобы установить один из параметров окружающей среды выше нормальных рабочих пределов и проанализировать

его воздействие на схему. Для электронного оборудования применяются три вида таких испытаний — механические нагрузки, температурные нагрузки и электрические нагрузки.

### 3.4.1. Механическая нагрузка

Систему можно подвергнуть механическому воздействию путем постукивания, а также изгибания или скручивания схемных плат. Такая проверка часто применяется в тех случаях, когда перемежающийся отказ из-за окислившегося краевого разъема, плохого контакта ИС в панельке или потускневшего вывода ИС вызывает хаотичное поведение системы. Тестирование нагрузкой призвано временно либо улучшить, либо ухудшить проявление отказа, что в любом случае помогает установить его причину. Тончайший разрыв печатного проводника, видимый только через увеличительное стекло, при скручивании платы либо дает соединение, либо становится шире. Скручивание и изгибание платы заставляют отказ появляться или исчезать, а это является признаком либо разрыва печатного проводника, либо наличия в панельке ИС потускневшего или отошедшего контакта. Потускневшие контакты легко очищаются таким известным средством, как карандашная стиральная резинка.

Конечно, применять механические нагрузки следует весьма осторожно, так как чрезмерное усердие может привести к появлению новых отказов. Однако способы «калиброванного кулака», скручивания и изгибания плат не отвергаются и при умелом применении часто могут указать тип отказа в системе и его возможное местонахождение.

### 3.4.2. Температурная нагрузка

Компоненты иногда выходят из строя из-за постепенного нагрева при внутреннем отказе, но до некоторого температурного предела работают исправно. Система с таким компонентом работоспособна до тех пор, пока микросхема не перегреется. Нагревшуюся микросхему легко обнаружить путем прикосновения. Временно перегрев можно устранить, охладив подозреваемую микросхему с помощью охлаждающего пульверизатора. В дру-

гом случае микросхема может работать на пределе, и небольшое повышение температуры частично или полностью выводит ее из строя. Такой режим можно получить искусственно с помощью фена, повысив температуру до появления отказа.

Температурное воздействие заключается в изменении температуры окружающей среды сверх обычных рабочих значений, чтобы вызвать отказ или временно отдалить его. Охлаждение с помощью пульверизатора оказывается более локальным, так как поток горячего воздуха от фена гораздо шире. Выявить на плате микросхему, которая работает с повышенной температурой (гораздо горячее остальных), можно путем касания, хотя некоторые микросхемы даже при нормальной работе нагреваются сильнее, чем можно предположить. Микросхемы одного и того же типа рассеивают различную мощность из-за разной рабочей частоты, поэтому сравнение их относительных температур может привести к неверным выводам. Дефектная микросхема может даже обжечь пальцы, поэтому касаться микросхем нужно осторожно.

#### 3.4.3. Электрическая нагрузка

Электрическое воздействие заключается в изменении подаваемых в систему напряжений либо для определения ее рабочего диапазона, либо для локализации микросхем, работающих на пределе. Пользоваться этим способом следует с большой осторожностью, так как перегрузка может вызвать катастрофические отказы многих компонентов. В тщательно спроектированной системе такое испытание не требуется, так как в ней на все микросхемы подается номинальное напряжение в пределах допусков, определенных в спецификациях. Наиболее вероятной причиной работы микросхемы на пределе является понижение напряжения питания до нижнего допустимого значения. Необходимо измерить напряжение питания в работающей системе и привести его к номиналу.

### 3.5. Локализация отказов

До выявления причин любого отказа его необходимо изолировать, локализовать с точностью до конкретной секции системы. В этом смысле поиск неисправности можно рассматривать как способ «сверху вниз», где

вначале имеется неисправная система и она расчленяется на подсистемы, в одной из которых находится отказ. В зависимости от размера системы для локализации отказа может потребоваться разделить каждую подсистему на меньшие подсистемы. Очевидно, что способ локализации отказа зависит от конструкции системы. Предпринимаемый подход будет различным для однопалатной системы и для системы, которая распределена по печатным платам в нескольких стойках.

Так как испытание и проверка даже простых вещей могут оказаться безуспешными, то будут иметь большое значение индивидуальное мастерство, интуиция и знание изделия. Воспользуйтесь всеми встроенными средствами локализации отказа (удаление печатных плат, служебные связи и перемычки и специальные тест-режимы). Целесообразно расчленить микропроцессорную систему, с тем чтобы иметь возможность диагностировать каждую часть в отдельности. Интерфейсные схемы и периферийные аналоговые схемы, применяемые для формирования сигналов, часто имеют большую интенсивность отказов, чем цифровые схемы системы. Объясняется это более высокими требованиями по быстродействию и рассеиваемой мощности, более высокими рабочими температурами, чувствительностью, точностью, временным и температурным дрейфом регулировок, внешними перегрузками и меньшими диапазонами безопасной работы. Сложные аналоговые микросхемы с высокой плотностью упаковки, используемые в интерфейсе микрокомпьютера, часто работают вблизи предельных значений параметров для достижения максимальной производительности. Интерференция сигналов в линиях синхронизации с шинами электропитания ТТЛ-схем часто вызывает серьезные проблемы наведенных помех.

Когда подозрение в отказе падает на цифровую часть системы, прежде всего следует убедиться в наличии сигналов. Пользуясь подходящими приборами, нужно проверить наличие сигналов синхронизации, сигналы на шине данных, сигналы в линиях разрешения работы микросхем и линиях шины управления. Отсутствие активности в любом из перечисленных узлов свидетельствует о возможном отказе. Наиболее частый отказ в цифровой ИС — разрыв во входном или выходном выводе. Когда подозревается такой отказ, целесообразно изолировать вывод от остальных схем. Быстрый и неразрушаю-

щий способ изолирования заключается в том, чтобы отсосать припой из отверстия, в котором находится вывод, с помощью специального приспособления. После этого вывод отгибается в центр отверстия, чтобы он нигде не касался медного проводника. Конечно, эта процедура довольно затруднительна, но вывод можно просто вытащить из отверстия. После этого вывод следует проверить на наличие сигналов; при подозрительном выходном выводе рекомендуется закорачивать его площадку на печатной плате на напряжение питания $V_{cc}$ или на землю и проверять схемы в тракте распространения сигнала.

Способы локализации цифровых блоков в изделии с микропроцессором сильно зависят от его электрической и механической архитектуры. Если некоторые печатные платы можно удалить с сохранением работоспособности базовой системы, этой процедурой можно воспользоваться для контроля функционирования частей системы. В системе может быть предусмотрен режим свободного счета, позволяющий проверить работу ядра системы по активности шины адреса и работу ПЗУ.

Плату удлинителя с переключателями на сигнальных линиях и шинах можно использовать для разрыва выбранных сигнальных трактов между отдельной схемной платой и остальной частью системы. Таким способом устраняются тракты обратной связи и «зависшие» шины. Еще более простой способ разрыва сигнальных трактов, проходящих через краевой разъем, заключается в том, чтобы поместить в нужные контакты разъема кусочек изолирующей ленты.

Отыскивать неисправности в петлях цифровой обратной связи довольно трудно, так как ошибки распространяются по этим петлям. В петле обратной связи с неправильным выходным сигналом он посылается на вход, что в свою очередь приводит к получению неверного сигнала на выходе. Разрыв тракта обратной связи предотвращает такое действие, и, следовательно, при задании управляемых сигналов на входах можно проследить их прохождение со входа на выход. Однако зачастую не так легко обеспечить подачу требуемых входных сигналов, так как при этом требуется одновременно управлять большим числом линий, связанных с проверяемым входом.

Иногда микросхемы удается динамически проверить

*на месте* с помощью *цифрового компаратора*. Прибор подключается к подозреваемой микросхеме, и в него вставляется заведомо исправная микросхема, аналогичная проверяемой. После этого компаратор сравнивает работу двух микросхем и индицирует несовпадение сигналов. Цифровой компаратор позволяет проверить большинство ТТЛ-микросхем в рабочих условиях, но, конечно, для каждой используемой микросхемы требуется аналогичная запасная.

Методы и средства поиска неисправностей, применяемые в изделиях, реализованных на основе так называемой «произвольной логики», малоэффективны для микропроцессорных систем, так как для получения содержательной информации о работе компьютера необходимо одновременно анализировать большое число сигналов. Для эффективного сокращения времени на локализацию и устранение отказа в систему встраиваются такие средства, как схемы контроля питания, диагностические программы, индикаторы правильного функционирования системы и сигнатурный анализ.

Если схемные платы в системе можно легко удалять и заменять на заведомо исправные, следует воспользоваться этой возможностью и изолировать отказ до одной схемной платы. Когда в системе применяются несколько идентичных схемных плат, например платы памяти в системе с шиной S100, то для проверки можно поменять их друг с другом. Конечно, здесь имеется риск повредить исправную плату, если отказ является следствием, а не причиной. При этом в заменяющей плате возникают те же проблемы, что и в первой, дефектной плате. Во всех случаях замену плат необходимо проводить с выключенным питанием, чтобы избежать повреждений микросхем из-за электрических перегрузок.

При наличии идентичной системы для локализации отказа производится функциональное сравнение. Оно полезно в том случае, когда неясно, существует ли отказ, — на самом деле это может быть особенность производства или конструктивное ограничение.

Если подозрение падает на микросхему в панельке попробуйте до замены покачать ее, чтобы устранить плохие контакты. Последней подозреваемой микросхемой является микропроцессор, но часто именно его стараются заменить прежде всего. Интенсивность отказов микропроцессоров очень низка; однако они являются

сложными приборами, и проверить их работу затруднительно. Такая же ситуация характерна для всех БИС, применяемых в микропроцессорных системах. В случае опытного образца или новой модели микросхемы может возникнуть явление, называемое «нехарактерным поведением», — микросхема, имеющая параметры, близкие к предельным, случайно выходит за пределы и начинает работать хаотически. Отказ такого типа очень трудно локализовать, так как он возникает только в определенных ситуациях и даже не всегда.

### 3.6. Дерево поиска неисправностей

Локализация отказов значительно упрощается при наличии тщательно продуманной последовательности тестов. Многие из этих тестов являются простым визуальным контролем, который может дать ценную информацию о возможном местонахождении отказа.

Дерево поиска неисправностей напоминает блок-схему и призвано как можно быстрее определить возможное местонахождение отказа. Действия, описываемые в каждом из элементов дерева, значительно варьируются от простого визуального осмотра до автономной и законченной тест-процедуры целой секции системы. Дерево поиска неисправностей можно разработать для любой системы (если его нет в документации), особенно после приобретения практического опыта работы с системой и в поиске неисправностей. После некоторого времени эксплуатации систему можно оснастить средствами контроля, которые учитывают наиболее распространенные и вероятные отказы. В качестве таких средств могут появиться переключатели для отсоединения секций системы и контрольные точки для удобного подключения разнообразных приборов. Дерево поиска неисправностей следует модифицировать с учетом дополнительных средств и корректировать его под все изменения, производимые в системе в течение ее срока службы.

Дерево поиска неисправностей проходится до локализации отказа, но необходимо подчеркнуть, что, как и в любой другой процедуре поиска неисправностей, исследователь должен осмыслить возможную причину отказа, а не слепо идти по ветвям дерева. Поиск неисправностей особенно в сложных изделиях напоминает игру вероятностей, когда перед продолжением теста необходимо ло-

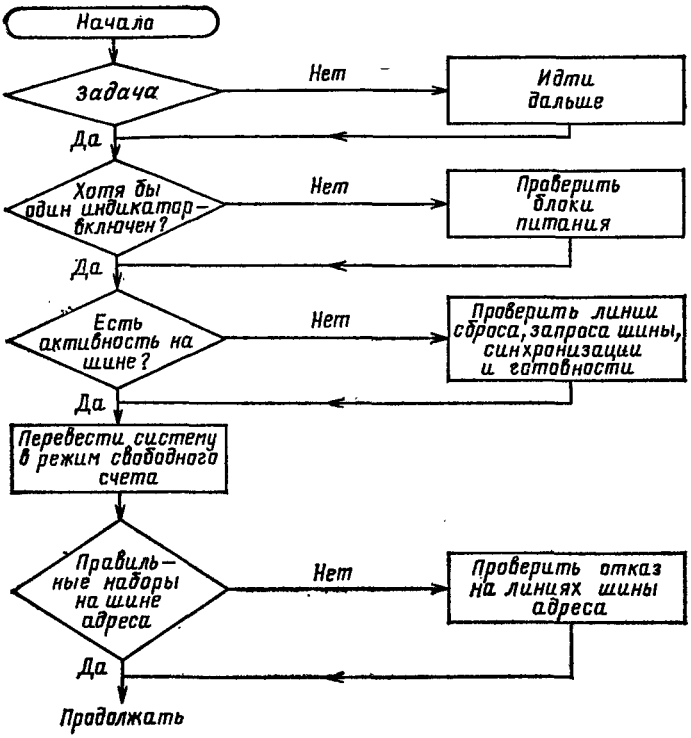



**Рис. 3.2.** Дерево поиска неисправностей

гически вывести наиболее вероятную причину отказа. Некоторые отказы, например разрыв вывода ИС, можно считать наиболее вероятной причиной неисправности; отсюда принимается решение о замене данной ИС для проверки правильности диагноза. На рис. 3.2 приведено начало дерева поиска неисправностей для небольшой системы, в которой имеется режим свободного счета.

Каждый ромб на дереве поиска неисправностей заключает в себе вопрос о поведении системы, начиная с очень простых вопросов, например, о том, включен хотя бы один индикатор. По мере движения по дереву методы и аппаратура, необходимые для ответа на вопросы, все более усложняются. Если, например, на вопрос «включен хотя бы один индикатор?» был получен отрицательный ответ, дерево показывает необходимость про-

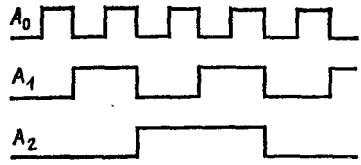



**Рис. 3.3.** Сигналы на шине адреса в режиме свободного счета

верки блока питания, для чего применяется цифровой вольтметр. Ответ на вопрос «шина адреса функционирует правильно?» потребует от исследователя применить более сложный прибор, чем цифровой вольтметр. Тест свободного счета заключается в подаче в микропроцессор некоторой разновидности холостой команды, так что он периодически формирует на шине адреса все возможные адреса, при этом, начиная со старшей линии адреса, частота сигналов на линиях адреса удваивается, и для проверки правильности работы шины адреса можно воспользоваться частотомером. Если, например, на двух линиях адреса обнаруживается одна и та же частота, следует предположить короткое замыкание между ними.

Как показано на рис. 3.3, для трех младших линий адреса в режиме свободного счета эти сигналы представляют собой как бы выходы двоичного счетчика, причем частота сигналов на каждой последующей линии в 2 раза меньше, чем на предыдущей. Другой способ проверки правильности работы шины рассматривается в гл. 7, посвященной сигнатурному анализу.

Дерево поиска неисправностей обеспечивает метод систематического поиска отказов в системе, начиная с общих отказов и кончая отказами микросхем или выводов. При прохождении дерева описывается возможная область и предлагаются необходимые тесты. Вместе с деревом поиска неисправностей требуются хотя бы краткие описания выполняемых тестов и их реализации в системе. Если такой информации нет, исследователю потребуется несколько справочников, чтобы определить, на каком выводе микросхемы действует контролируемый сигнал.

Поясняющая информация должна быть достаточно полной, чтобы исследователю не нужно было обращаться за необходимой информацией к другой документации. К сожалению, для большинства систем подобная информация, облегчающая работу исследователя, отсутствует.

Довольно часто исследователь должен сам строить дерево поиска неисправностей и просмотреть множество справочников и технических описаний, чтобы отыскать необходимую для этого информацию. Исторически сложилось, что изготовители вычислительных систем поставляют системы с недостаточно информативными руководствами и без диаграмм поиска неисправностей, которые обычно разрабатываются в результате усилий пользователей системы.

Чтобы компенсировать безразличное отношение изготовителей систем, несколько крупных корпораций пользователей определили свои принципы тестирования и заставили изготовителей аппаратуры и субподрядчиков составлять информацию о тестировании в соответствии с их спецификациями с целью сокращения простоя систем. Такой подход принят в военно-морских силах Великобритании, а также в корпорации British Steel, которая требует поставлять для каждой системы *функциональную системную документацию* (FSD), предназначенную для ускоренного контроля и ремонта системы. Пассивное отношение изготовителей систем к проблеме контроля понятно, так как подготовка полного руководства по тестированию оказывается длительной и трудной задачей, если система проектировалась без учета необходимости поиска неисправностей в ней. Сам пользователь не беспокоится об отсутствии такой информации до тех пор, пока система не выходит из строя и не возникает необходимость ее ремонтировать. Похоже, что изготовители и пользователи систем озабочены проблемами тестирования сложных вычислительных систем и начинают встраивать соответствующие средства, например режим свободного счета и диагностические программы, которые можно модифицировать по мере совершенствования самой системы.

## 4. Применение обычных приборов

Под обычной контрольно-измерительной аппаратурой мы понимаем те приборы, которые применялись для контроля электрических и электронных схем до появления в 1971 г. микропроцессора, а именно *осциллографы, цифровые вольтметры и частотомеры.* Хотя эти приборы широко применяются для общего тестирования

систем, в вычислительных системах их полезность ограничена характером представления информации — в параллельных кодах на большом числе линий одновременно. Однако структура вычислительной системы не отвергает их совсем, и имеется много примеров, когда обычная контрольно-измерительная аппаратура активно используется в тестировании. Мы уже приводили примеры ее использования — контроль напряжения электропитания и проверка частоты сигналов на линиях шины адреса в режиме свободного счета.

Рассматриваемые приборы имеются в большинстве отделов, связанных с тестированием, и персонал должен отчетливо понимать принципы их работы и области применения, хотя эти приборы играют вспомогательную роль в исследовании сложных цифровых вычислительных систем. Они должны дополняться специализированной аппаратурой, рассчитанной на обработку сложных форматов данных, которые характерны для вычислительных систем.

### 4.1. Мультиметры

Мультиметр — это прибор, предназначенный для измерения постоянных токов и напряжений, переменных напряжений и, часто, токов, а также сопротивлений. Устаревший и сравнительно простой мультиметр, по существу, представляет собой амперметр, который выполняет требуемые измерения при включении в схему, содержащую резисторы и источники электропитания. Во всех случаях в конце концов измеряется постоянный ток, приводящий в движение индукционные катушки амперметра. Типичный прибор содержит амперметр со шкалой 50 мкА, который часто характеризуется чувствительностью 20 000 Ом/В. Поэтому на шкале 3 В суммарное сопротивление в схеме с учетом сопротивления самого амперметра будет 60 000 Ом.

При работе в низковольтном диапазоне относительно малое сопротивление прибора представляет значительную нагрузку для измеряемой схемы, что приводит к неправильным показаниям. Новые *цифровые вольтметры* измеряют напряжение в схеме, имея высокое входное сопротивление, и не нагружают измеряемую точку электронной схемы. Основное измерение здесь связано с напряжением, а не с током, который может быть опреде-

лен только путем введения сопротивления последовательно с измеряемой цепью и измерения падения напряжения на нем. Типичное полное входное сопротивление цифровых вольтметров при измерении постоянных напряжений составляет 10 МОм. В большинстве современных цифровых вольтметров применяется *аналого-цифровой преобразователь,* а результат показывается на светодиодном или жидкокристаллическом индикаторе.

В цифровых системах вольтметры применяются для измерений напряжений питания и потребляемых токов. Например, стандартный $3\frac{1}{2}$-разрядный цифровой вольтметр на диапазоне $00{,}00 \div \pm 19{,}99$ В можно использовать для измерения напряжения питания 5 В, характерного для большинства логических схем. Преобразователи в цифровых вольтметрах осуществляют пять измерений в секунду, поэтому приборы нельзя применять для фиксации быстрых выбросов напряжения, которые возникают в системе при отказе. Это же относится и к измерению потребляемого тока, так как прибор показывает только среднее значение.

С помощью мультиметра можно найти короткие замыкания линий питания (в выключенной системе) из-за таких отказов, как короткие замыкания танталовых конденсаторов. После установления факта короткого замыкания мультиметр почти не помогает в локализации отказа, если не пользоваться последовательной отпайкой конденсаторов. При отсутствии других приборов такой прием оказывается единственно возможным для исследователя, хотя, как мы увидим в гл. 5, имеются более быстрые и эффективные способы локализации отказа подобного типа.

В большинстве современных цифровых вольтметров предусматривается режим контроля целостности — при наличии соединения между проверяемыми точками схемы появляется звуковой сигнал. Это средство очень удобно при проверке целостности кабелей и печатных проводников.

### 4.2. Частотомеры

Основу частотомера составляет управляемый по входу счетчик, в который входной сигнал подается в течение временно́го интервала определяемого точным и стабильным генератором. Если, например, схема генерато-

ра разрешает прохождение входного сигнала в течение 1 с и счетчик был установлен в нуль, содержимое счетчика будет равно средней частоте повторения импульсов входного сигнала. Счетчик имеет ограниченную разрядность, поэтому для измерения широкого диапазона входных частот временной интервал может изменяться. Для измерения высокой частоты интервал устанавливается коротким, а в случае низкочастотного входа он соответственно увеличивается.

Частотомер применяется для измерения частоты генератора синхронизации компьютера, которая определяет все его операции и все другие производные частоты в системе. Частоты для управлений строчной и кадровой развертками дисплея с *электронно-лучевой трубкой* (ЭЛТ) получаются из основной частоты с помощью делителей, и их можно измерить частотомером. При передаче информации по последовательным линиям скорость передачи во многих системах определяется частотой, полученной из основной частоты системной синхронизации, и ее также можно измерить с помощью частотомера.

В большинстве вычислительных систем применяются генераторы с кварцевой стабилизацией, и их частота редко изменяется до таких пределов, чтобы вызвать в системе серьезные проблемы, поэтому при тестировании вычислительных систем частотомер требуется редко. Частотомеры могут быть использованы и для измерения временных интервалов. Это позволяет определять с их помощью продолжительность управляющего сигнала, посылаемого в выходной порт, и, следовательно, использовать частотомер как средство проверки правильности программы, которая формирует этот сигнал.

Некоторые приборы имеют в своем составе цифровой вольтметр и измеритель частот/интервалов, а также другие средства цифрового тестирования.

### 4.3. Осциллограф

Осциллограф отображает электрические сигналы в виде осциллограммы на ЭЛТ. Типичный двухканальный осциллограф позволяет одновременно наблюдать на экране два сигнала, а протяженностью индикации во времени можно управлять посредством встроенного в прибор генератора. Рассматриваемый нами базовый прибор удобен для фиксации и индикации периодических сигна-

лов независимо от их формы. Схема запуска в осциллографе устанавливается на инициирование развертки, когда входной сигнал превышает заданное напряжение, а направление запуска (т. е. сигнал с положительным или отрицательным переходом) можно выбирать. В случае периодического входного сигнала после установки осциллографа на запуск в требуемой точке и после установки скорости развертки на экране появляется неподвижное изображение, так как последовательные сигналы накладываются друг на друга. Послесвечение люминофора экрана ЭЛТ достаточно велико, чтобы последовательные сигналы вызывали новое свечение до того, как затухнет след предыдущего сигнала.

Для фиксации и индикации непериодических сигналов применяются запоминающие осциллографы, которые при запуске воспринимают входной сигнал и индицируют его либо до естественного затухания, либо до стирания с целью воспроизведения нового сигнала. В старых моделях запоминающих осциллографов применялось специальное покрытие экраиа, которое могло хранить электрический заряд некоторое время до его исчезновения из-за токов утечки. В новых запоминающих осциллографах входной сигнал преобразуется в цифровую форму, и воспринятые отсчеты хранятся в ЗУПВ. Затем они преобразуются в аналоговую форму и индицируются на обычной ЭЛТ. Хранимые в ЗУПВ цифровые значения не подвержены затуханию, и изображение сохраняется до выключения прибора. Основной недостаток подобных приборов связан с ограниченной емкостью ЗУПВ, которая определяет и число уровней квантования, и число хранимых отсчетов сигнала. Конечно, по мере появления все более емких полупроводниковых ЗУПВ этот недостаток становится менее критичным.

В вычислительной системе при выполнении программы нет периодических сигналов, поэтому при индикации на осциллографе видны почти случайные последовательности двоичных сигналов, которые переходят из одного логического состояния в другое в произвольные моменты времени. Осциллограф нельзя запускать такими сигналами из-за нерегулярных изменений состояния, и требуется какой-либо синхронизирующий сигнал, например от системного генератора синхронизации. Двухканальный осциллограф индицирует всего два сигнала, поэтому как регистрирующий прибор он почти бесполезен, ес-

ли, например, требуется проанализировать шину данных. Осциллограф показывает процесс на отдельных линиях шины адреса или шины данных, однако ограниченный объем получаемой при этом информации препятствует принятию осмысленных решений.

При определенных условиях осциллограф можно применить для проверки блоков вычислительной системы. В § 2.5 описан метод свободного счета для проверки ядра системы; здесь осциллографом измеряется частота повторения сигналов на всех линиях шины адреса, чтобы убедиться в отсутствии отказов. Тест свободного счета формирует периодические сигналы на всех линиях адреса, и запуск осциллографа возможен с любой из них. С помощью простых тест-программ образуются периодические сигналы в других блоках вычислительной системы, иапример в выходных портах.

Рассмотрим простую тест-программу.

| Мнемоники Z80 | | | Мнемоники 8085 | | |
|---|---|---|---|---|---|
| | LD | A, 00H | | MVI | A, 00H |
| LOOP: | OUT | (01), A | LOOP: | OUT | 01 |
| | INC | A | | INR | A |
| | JP | LOOP | | JMP | LOOP. |

В приведенных фрагментах использованы мнемонические коды микропроцессоров Z80 и 8080/8085. Вначале регистр *A* в ЦП сбрасывается в нуль, а затем это значение посылается в выходной порт с адресом 01. После этого производится инкремеит регистра *A*, и программа переходит к посылке нового значения в тот же самый выходной порт. В программе реализован бесконечный цикл, поэтому выводимые в порт данные проходят последовательность от 00 до FF, осуществляется сброс на 00, и счет начинается вновь. С помощью осциллографа проверяются все 8 линий выходного порта.

При поиске неисправностей в цифровых схемах часто приходится одновременно контролировать множество логических элементов. Проблема здесь заключается в том, что большинство логических элементов имеют два и более входов и один выход, сигналы на которых в общем случае необходимо анализировать одновременно. Стандартный двухканальный осциллограф одновременно показывает только два сигнала. С помощью расширителей можно расщепить каждый вход *Y* с индикаци-

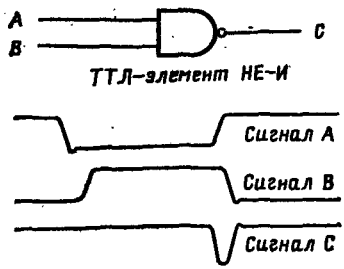



**Рис. 4.1.** Помеха, вызываемая гонками

ей в одном канале двух сигналов, что позволяет индицировать четыре сигнала, но для достижения этого требуется коммутировать каждый входной канал с одного сигнала на другой. В такой системе возникает следующая проблема: для индикации события, длящегося 10 нс, частота коммутации канала должна быть более 200 МГц. Схемы, работающие на столь высокой частоте, оказываются критичными к топологии и весьма дорогими. В некоторых осциллографах вместе с двумя обычными входами индицируется запускающий сигнал, что позволяет одновременно наблюдать три сигнала.

На этапах разработки и опытного производства схема элемента рассчитывается на одновременное появление двух входных сигналов, но из-за небольших различий задержек распространения один из входных сигналов появляется раньше другого. Такая ситуация называется «гонкой», и она является причиной явления, показанного на рис. 4.1. Два входных сигнала в элементе НЕ-И подвержены «гонкам», и один из них изменяется раньше другого, что приводит к появлению на выходе выброса или узкой помехи (glitch). В зависимости от типа элемента продолжительность помехи может составлять всего несколько наносекунд, но этого бывает достаточно для запуска последующей схемы с появлением хаотичного поведения. Для обнаружения данного явления удобно использовать осциллограф, который показывает три сигнала одновременно.

Аналогичная ситуация возникает в системе в основном из-за ошибок разработчика, причем она может не проявиться как отказ. Когда же позднее система расширяется и сигналы с помехами начинают использоваться в других подсистемах, они приводят к непредсказуемому поведению системы.

На рис. 4.2 показан 3-битный синхронный счетчик, реализованный на JK-триггерах 7470. Предположим, что он должен считать до 4, а затем сбрасываться в нуль и

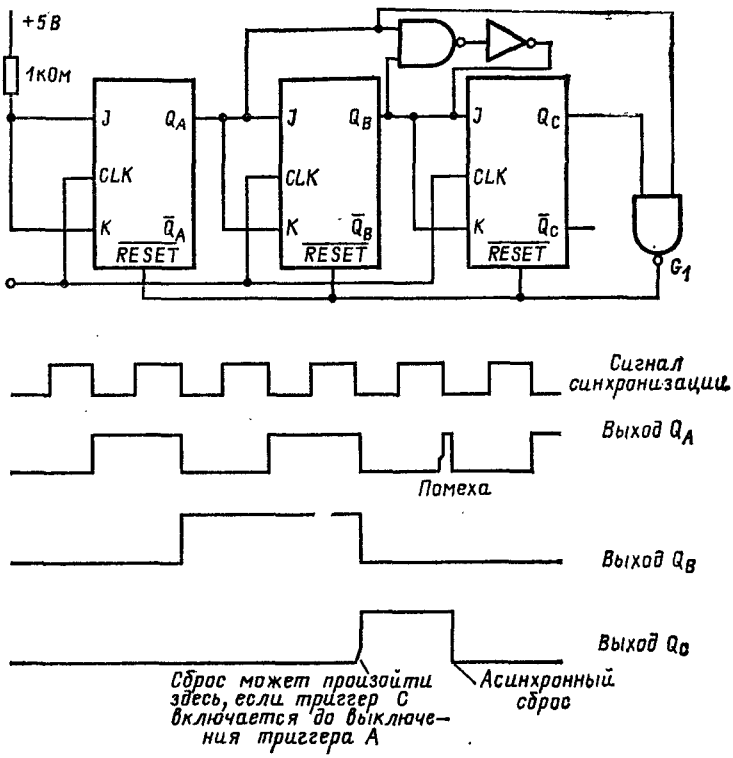



**Рис. 4.2.** Помеха, вызываемая плохой проработкой

вновь начинать счет. Асинхронная схема сброса фиксирует двоичный код 5 и вызывает сброс счетчика в нуль. Подчеркнем, что приведенная схема служит примером плохой разработки и иллюстрирует те проблемы, которые появляются из-за этого. Продолжительность помехи, которая видна на выходе $Q_A$, определяется задержками распространения в элементе НЕ—И схемы сброса и в триггерах, когда они сбрасываются в нуль по входам сброса. В стандартных ТТЛ-схемах продолжительность помехи будет не более 68 нс, что при периоде синхронизации 1 с составляет 0,0000057 % одного периода. Если выходы счетчика подключены на светодиодные индикаторы, они не отреагируют на помеху и будут показывать правильные состояния 0...4. Когда же позднее выходы

счетчика подадутся в другую схему, помеха может вызвать ее неправильные действия. Такую короткую помеху во множестве сигналов, изменяющих состояния с намного меньшей частотой, очень трудно обнаружить осциллографом, который должен запускаться сигналом с самой низкой частотой повторения. Для решения подобного рода проблем разработаны специальные анализаторы, которые могут зафиксировать очень короткие асинхронные сигналы.

### 4.4. Ограниченные возможности обычных приборов

Обычные контрольно-измерительные приборы, которые применяются для решения некоторых общих проблем, возникающих в электронном оборудовании, играют ограниченную роль при поиске неисправностей в системах с шинной структурой, например в микрокомпьютерах. Параллельное представление информации на многих линиях одновременно и высокая скорость изменения информации приводят к тому, что обычные приборы не помогают в решении проблем, возникающих в вычислительных системах. Стремительное развитие микропроцессорных систем поставило вопрос о разработке специальной контрольно-измерительной аппаратуры, ориентированной на такие системы. Эта аппаратура появилась не сразу и варьируется от простых ручных инструментальных средств до сложных анализаторов, которые непрерывно модифицируются и совершенствуются.

## 5. Ручные инструментальные средства

По-видимому, первыми образцами приборов, специально предназначенных для поиска неисправностей в цифровых схемах, были логический пробник, логический пульсатор, индикатор тока и логический компаратор. За исключением логического компаратора, они применяются либо для возбуждения, либо для контроля отдельных узлов в логической системе и помогают определить логическое состояние узла и его работоспособность. Ручные средства применяются в отдельности при проверке системы или совместно для реализации

проверок по принципу «стимул—реакция». Обычно их питание осуществляется от проверяемой системы, а конструктивно ручные средства оформлены как небольшие и очень удобные приборы.

Кроме перечисленных логического пробника, пульсатора и индикатора тока имеются логические клипсы и компараторы. Они применяются для функционального контроля одной микросхемы при работе ее в системе. Логическая клипса надевается на проверяемую ИС и получает питание от вывода самой ИС. Логическая схема внутри клипсы определяет полярность питания, а светодиоды на торце клипсы показывают логические состояния выводов ИС. Логическая клипса может проверять одно логическое семейство, например ТТЛ, и даже с ограничениями внутри семейства из-за большого разнообразия способов подключения питания и значительного числа типов корпусов ИС. Например, клипса может проверять ИС в корпусах типа DIP (с двусторонним расположением выводов), имеющих 14 или 16 выводов. Даже среди микросхем с такими корпусами клипса может проверять не все микросхемы. Внутри клипсы имеется схема с довольно ограниченными возможностями, поэтому быстрые импульсные события на выводах проверяемой микросхемы нельзя видеть на светодиодах, индицирующих состояния этих выводов. Большинство ограничений, свойственных логической клипсе, устранено в логическом компараторе, который воспринимает сигналы от проверяемой микросхемы через пассивную клипсу и плоский кабель. В компаратор помещается ИС, аналогичная проверяемой, и любые различия в работе двух микросхем индицируются на светодиодах. Обе микросхемы работают параллельно, но выходы микросхемы, находящейся в компараторе, действуют только в самом компараторе для получения и последующей индикации сигналов «правильно/неправильно». Обычно логические компараторы оснащаются платой персонификации для каждой проверяемой микросхемы; эта плата настраивает прибор и дает информацию о выводах питания, входах и выходах. Логический компаратор универсальнее логической клипсы и может проверять большинство микросхем семейства элементов при наличии панелек для эталонных микросхем и кабелей для разных типов корпусов.

## 5.1. Логические пробники

Логический пробник контролирует поведение одной точки в системе и с помощью нескольких индикаторов сообщает пользователю о том, находится проверяемая точка в состоянии логической 1, состоянии логического 0 или имеет промежуточный уровень. Большинство пробников показывают наличие также импульсов в точке схемы вспышками одного из индикаторов. Для показа логического состояния точки применяются либо отдельные индикаторы, либо один индикатор, который ярко светится в состоянии логической 1 и выключен в состоянии логического 0. Если проверяемый узел имеет искаженный логический уровень (завышенный 0 или заниженная 1), единственный индикатор светится вполнакала.

Уровни напряжения, соответствующие логическим сигналам, задаются не фиксированными потенциалами, а диапазонами напряжений.

Номинальное рабочее напряжение ТТЛ-схем составляет 5 В. Диапазоны напряжений, которые представляют два логических состояния на входе и выходе схемы, показаны на рис. 5.1. Когда выходным состоянием ТТЛ-схемы является логический 0, фактическое напряжение на биполярном транзисторе, закорачивающем выходную цепь на землю, зависит от значения отводи-

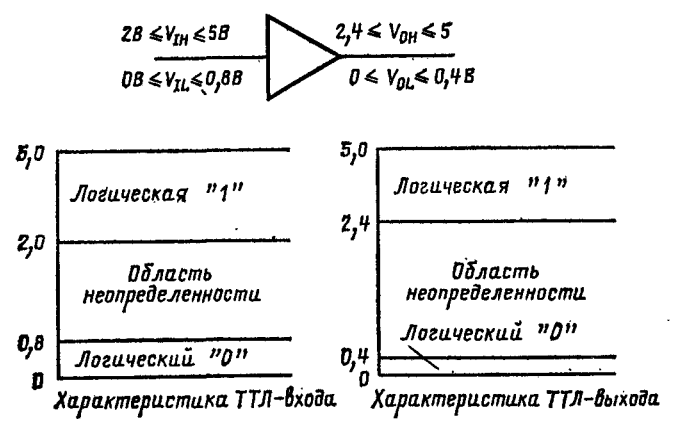



**Рис. 5.1.** Характеристики входиого и выходного напряжений в ТТЛ-элементе

мого на землю тока. Для ТТЛ-схем максимальное гарантированное напряжение на выходном транзисторе составляет 0,4 В при условии, что число входов, подключенных к данному выходу, не превышает заданного предела. Этот предел называется коэффициентом разветвления элемента по выходу, и для стандартных ТТЛ-схем он равен 10, т. е. выход одного ТТЛ-элемента допускается подключить максимум к 10 входам ТТЛ-элементов.

ТТЛ-схемы имеют запас помехоустойчивости из-за того, что стандартный ТТЛ-вход воспринимает напряжение до 0,8 В как уровень логического 0. Аналогично минимальный выходной уровень логической 1 составляет 2,4 В, а вход воспринимает как уровень логической 1 напряжение более 2 В.

Для ТТЛ-элемента, представленного на рис. 5.1, определяются следующие напряжения:

$V_{IH}$ — входное напряжение внутри диапазона, отведенного для представления состояния логической 1; обычно указывается минимальное значение, т. е. 2 В для стандартных ТТЛ-схем;

$V_{IL}$ — входное напряжение внутри диапазона, отведенного для представления логического 0; обычно определяется максимальное значение, т. е. 0,8 В;

$V_{OH}$ — напряжение высокого уровня на выходе; обычно задается минимальное значение, т. е. 2,4 В;

$V_{OL}$ — напряжение низкого уровня; обычно указывается максимальное значение, т. е. 0,4 В.

Разности между гарантированными входными и выходными высоким и низким уровнями определяют возможности элемента подавлять возникающие в системе помехи.

Влияние помех можно показать на простой схеме из двух элементов, приведенной на рис. 5.2. Предположим, что выход элемента $G_1$ формирует низшее гарантированное напряжение $V_{OH}$ высокого уровня, т. е. имеет значение 2,4 В. Источник помех, представленный на рис. 5.2 идеальным генератором напряжения $V_N$, включен последовательно с выходом элемента $G_1$, поэтому на-

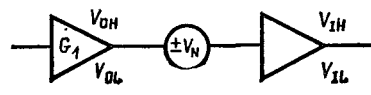



**Рис. 5.2.** Два ТТЛ-элемента, соединенных линией с помехами

пряжение на входе элемента $G_2$ равно сумме $V_{OH}$ и $V_N$:

$$V_{IH} = V_{OH} \pm V_N.$$

Напряжение $V_{IH}$ для гарантированной работы элемента $G_2$ должно быть минимум 2 В, поэтому, если $V_N$ будет больше —0,4 В, входное напряжение $G_2$ упадет ниже гарантированно входного минимума и может вызвать хаотичное поведение $G_2$. Значение $V_N$, которое вызывает такое нежелательное явление, называется границей помех, и в случае ТТЛ-схем оно равно 0,4 В. Другими словами, ТТЛ-схемы допускают уровни наводимых помех 0,4 В и менее, прежде чем будут превышены гарантированные рабочие параметры. Аналогично, если выход $G_1$ имеет низкий уровень, входное напряжение элемента $G_2$ равно:

$$V_{IL} = V_{OL} \pm V_N.$$

Если $V_{OL}$ имеет максимальное значение 0,4 В и $V_N$ превышает 0,4 В, напряжение $V_{IL}$ будет больше максимального значения 0,8 В, и элемент будет работать вне гарантированных пределов. Граница помех в состоянии логического 0 равна 0,4 В. Следовательно, ТТЛ-схемы в любом логическом состоянии допускают уровни помех до 0,4 В, прежде чем будут превышены их рабочие параметры.

Между максимальным напряжением логического 0 и минимальным уровнем логической 1 имеется диапазон напряжений, в котором работа схемы не определена. Обычно при переходе из одного логического состояния в другое вход или выход быстро проскакивает эту область неопределенности. Если, например, внутренний проводник, соединяющий кристалл с выводом корпуса, оборван, все подключенные к этому выводу входы будут иметь напряжение внутри области неопределенности, что в предельном случае может вызвать генерацию.

Логический пробник для исследования ТТЛ-схем должен различать три возможных состояния схемы — 1, 0 и напряжение внутри области неопределенности. Для достижения универсальности пробник должен проверять и КМОП-схемы, логические пороги которых отличаются от порогов ТТЛ-схем. В отличие от ТТЛ-схем, работающих с фиксированным напряжением питания 5 В, КМОП-схемы могут работать при напряжении питания в диапазоне 3—18 В. Пороговые уровни и границы

помех обычно задаются в долях номинального напряжения питания $V_{dd}$, причем типичное значение для $V_{IL}$ равно 0,3 $V_{dd}$, а для $V_{IH}$ — 0,7 $V_{dd}$. В статическом режиме выходные пороги являются почти идеальными: 50 мВ для $V_{OL}$ и на 50 мВ ниже $V_{dd}$ для $V_{OH}$. Типичное значение границы помех для КМОП-схем составляет 45 % от $V_{dd}$, что намного больше, чем у ТТЛ-схем. Это объясняет выбор КМОП-схем для применения в условиях с высокими уровнями помех. Однако полное выходное сопротивление КМОП-схем примерно в 10 раз больше, чем у ТТЛ-схем, а это делает нх более подверженными наводимым помехам, хотя высокая помехоустойчивость КМОП-схем ие приводит к серьезным проблемам.

КМОП-схемы часто предпочитают из-за малого потребления энергии по сравнению с ТТЛ-схемами, что особенно важно при питании от аккумуляторов. Логические элементы работают в динамических схемах, где происходят частые переключения состояний. Потребление энергии КМОП-элементов быстро увеличивается с ростом частоты переключения и на частотах в диапазоне 500 кГц — 2 МГц превышает потребление энергии эквивалентных маломощных ТТЛШ-элементов. Это обстоятельство необходимо учитывать при проектировании систем, рассчитаиных на малое потреблеиие энергии, но с высокой частотой переключения, так как КМОП-схемы могут проиграть по сравнению с другими логическими семействами.

Универсальный логический пробник должен работать с ТТЛ- и КМОП-схемами, хотя во многих системах, построенных на основе КМОП-схем, применяется питание 5 В. Пробник должен индицировать логические состояния ВЫСОКИЙ и НИЗКИЙ, а также «плохие» уровни, включая выходы с разрывом и выходы типа открытого коллектора без нагрузочных резисторов. Кроме индикации статических состояний пробник должен показывать пользователю также динамическое поведение узла.

Пробник должен индицировать не только наличие коротких импульсов с высокой частотой повторения, но и прохождение одиночных коротких имнульсов. Оба вида динамических сигналов индицируются посредством расширения любого короткого импульса до 50—60 мс, которых достаточно, чтобы пользователь заметил изменение состояния индикатора с «выключено» на «включено» и наоборот.

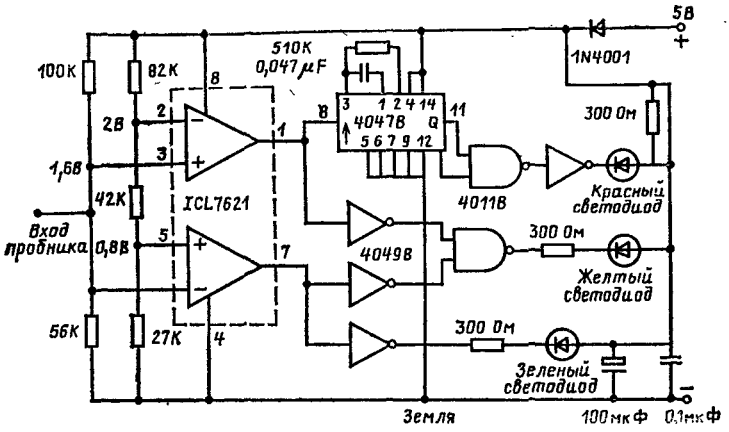



**Рис. 5.3.** Простой логнческий пробник

Логические пробники варьируются от простых устройств до сложных приборов, содержащих специально для них разработанные микросхемы. На рис. 5.3 показана схема логического пробника, предназначенного для проверки ТТЛ-схем.

Питание пробника осуществляется через гибкий провод от проверяемой системы. Диод 1N4001 в цепи питания предотвращает повреждение при неправильном подключении питания. С помощью резисторного делителя, незначительно нагружающего узел, вход пробника смещен примерно на уровень 1,6 В, который находится в области неопределенности для ТТЛ-схем. Вход пробника подан на сдвоенный операционный усилитель, выполненный по КМОП-технологии, который сравнивает входное напряжение пробника с фиксированными напряжениями, полученными от делителя.

Выход верхнего операционного усилителя переходит из состояния логического 0 в состояние логической 1, когда входное напряжение превышает 2 В. Выход нижнего операционного усилителя изменяется из состояния логического 0 в состояние логической 1, когда входное напряжение пробника ниже 0,8 В. Если вход пробника свободен, выходы обоих операционных усилителей находятся в состоянии логического 0, что вызывает включение желтого светодиода, показывающего «плохой» уровень. Когда напряжение на входе больше 2 В, желтый

светодиод выключается, а красный светодиод светится, показывая уровень логической 1. Если входное напряжение ниже 0,8 В, светится только зеленый светодиод, показывая уровень логического 0. Любой импульсный сигнал в проверяемом узле вызывает изменение выхода верхнего операционного усилителя из одного логического состояния в другое. Положительный фронт сигнала от этого операционного усилителя запускает одновибратор 4047 В примерно на 60 мс, что дает вспышку красного светодиода. Таким образом, пробник индицирует статические логические уровни высокого, низкого и промежуточного уровней напряжения, а также прохождение импульсных сигналов в узле. КМОП-схемы обеспечивают малое потребление тока от проверяемой системы (около 12 мА). Вся схема пробника размещается в корпусе с размерами 30×20×140 мм. Основным недостатком схемы является узкая полоса пропускания — 500 кГц, которая ограничивается частотными характеристиками сдвоенных операционных усилителей ICL7621, и поэтому пробник не фиксирует помехи с длительностью в десятки и сотни наносекунд.

#### 5.1.1. Промышленные логические пробники

Несколько фирм предлагают логические пробники, которые могут обнаруживать одиночные импульсы длительностью до 10 нс и с частотой до 80 МГц. Коммутируемый пробник обеспечивает проверку схем, выполненных по технологиям ТТЛ, ДТЛ, РТЛ, МОП, КМОП и др. Примером логического пробника с такими возможностями служит пробник модели 545А фирмы Hewlett-Packard. Он расширяет одиночные импульсы продолжительностью 10 нс и более до 50 мс для индикации на газоразрядной лампе, размещенной в его зонде. Интенсивность свечения этого единственного индикатора информирует пользователя об одном из трех состояний узла.

На рис. 5.4 и 5.5 показаны реакции пробника модели 545А на различные входные сигналы ТТЛ- и КМОП-схем, работающих от напряжения электропитания 5 В. Пороговые уровни логической 1 и логического 0 находятся в небольших диапазонах напряжений около номинальных значений. Для ТТЛ-схем пороговый уровень логической «1» составляет $2^{+0,4}_{-0,3}$ В, а пороговый уровень

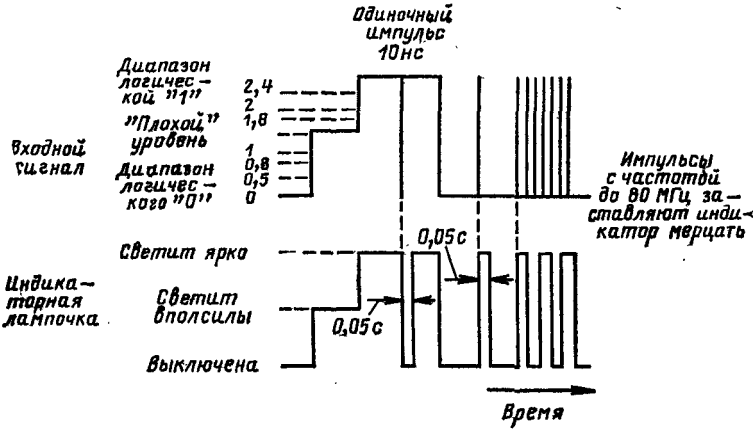



**Рис. 5.4.** Реакцин на ТТЛ-сигналы логического пробника 545А

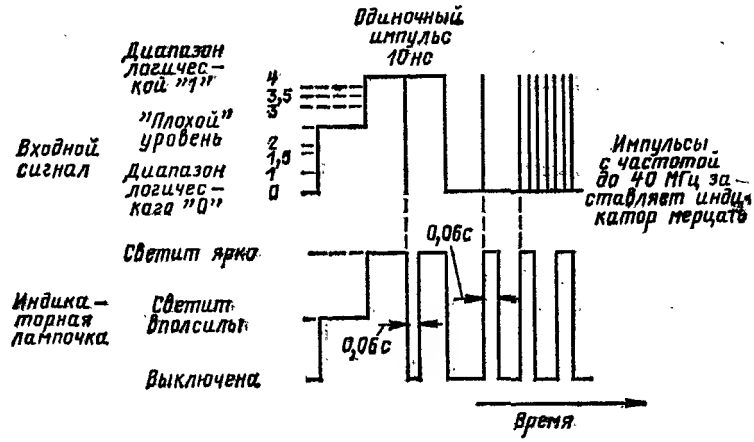



**Рис. 5.5.** Реакции на КМОП-сигиалы логического пробника 545А (питание 5 В)

логического 0 равен $0{,}8^{+0{,}2}_{-0{,}3}$ В. Напряжение питания КМОП-схем обозначается $V_{dd}$, и пороговые уровни пробника модели 545А определяются относительно $V_{dd}$. Порог логической 1 составляет $0{,}7\,V_{dd}\pm0{,}5$ В, а порог логического 0 равен $0{,}3\,V_{dd}\pm0{,}5$ В.

При испытании ТТЛ-схем пробник обнаруживает и индицирует одиночные отрицательные импульсы шириной до 10 нс с выключением индикатора на 50 мс. При

обнаружении положительных импульсов шириной до 10 нс индикатор вспыхивает на 50 мс. Последовательности импульсов с частотой до 80 МГц вызывают мерцание индикатора с частотой 10 Гц. При испытании КМОП-схем максимальная частота обнаруживаемых импульсов составляет 40 МГц, а одиночные импульсы шириной до 10 нс расширяются до 60 мс.

Пробник модели 545А обладает возможностью запоминания — светодиод, размещенный в корпусе пробника, включается, когда пробник обнаруживает изменение логического состояния. Имеющейся кнопкой можно сбросить этот индикатор, который вновь включится при обнаружении очередного изменения состояния. Индикатор удобен для фиксации одного или нескольких импульсов в тех случаях, когда неудобно наблюдать индикатор в зонде или когда рассматриваются редкие импульсы.

#### 5.1.2. Использование логического пробника

Главное назначение логического пробника — проверить подачу питания на микросхемы, проконтролировать статические уровни в логических элементах на правильность функционирования и установить наличие импульсов в проверяемых цепях.

С помощью логического пробника удобно проверить наличие питания на микросхемах. Почти во всех ТТЛ-элементах питание $V_{cc}=5$ В обычно подается на вывод *14* (корпус DIP с 14 выводами) или на вывод *16* (корпус DIP с 16 выводами). Земля обычно подается на выводы *7* и *8* соответственно. Впрочем, имеются и исключения из приведенных правил, например микросхема десятичного счетчика 7490, но в подавляющем большинстве ТТЛ-микросхем правила соблюдаются. Если коснуться зондом пробника выводов *14* и *16*, его индикатор при наличии питания ярко засветится. Конечно, пробник не показывает значения напряжения питания, а просто сигнализирует о наличии электропитания; фактическое значение напряжения питания можно измерить с помощью цифрового вольтметра. Когда же зонд пробника касается выводов *7* или *8*, индикатор должен быть выключен. Если в любом случае индикатор светится «вполнакала», следует предположить обрыв цепей, и необходимо просмотреть печатные проводники от выводов микросхемы к блоку питания.

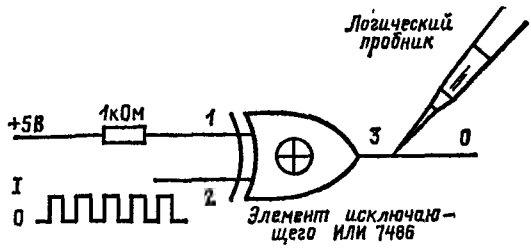



**Рис. 5.6.** Резервный элемент исключающего ИЛИ работает как инвертор

С помощью логического пробника можно проверить правильность функционирования логического элемента.

Когда место на печатной плате ограничено, вместо попыток разместить на ней дополнительные микросхемы часто оставляют незадействованными (резервными) логические элементы в работающих микросхемах. Рассмотрим, например, элемент исключающего ИЛИ на рис. 5.6. Микросхема 7486 содержит в одном корпусе четыре таких элемента, из которых, возможно, задействованы только три. Если в системе потребуется инвертор, его можно реализовать с помощью резервного четвертого элемента микросхемы. Касание логическим пробником вывода *1* покажет ярким свечением индикатора состояние логической *1*, а касание вывода *2* — наличие импульсов. Функция элемента должна быть такой, что на выходе должна получаться инвертированная входная последовательность, поэтому при касании пробником выхода также должно быть индицировано наличие импульсов. Тот факт, что последовательность импульсов инвертирована относительно входной, по индикатору логического пробника определить невозможно. Если вместо сигнализации о наличии импульсов на выходе элемента индикатор остается выключенным, то в схеме имеется отказ, которым может быть либо отказ в самом элементе, либо закорачивание на землю вне элемента. Короткое замыкание может быть вызвано либо неаккуратной пайкой, приводящей к соединению между линией с выхода элемента и землей, либо замыканием на землю входа внутри любой из микросхем, к которой подключен выход элемента. Для определения фактического отказа необходимо либо изолировать выходной вывод (см.

§ 3.5), либо с помощью индикатора тока установить, через какую микросхему течет ток.

В микропроцессорной системе логический пробник удобно применять для первоначального контроля статических логических уровней и проверки работоспособности шины. Следует проверить линии шины управления, чтобы убедиться в том, что отказ на одной из критических управляющих линий не препятствует работе системы, а это может случиться, если, например, на входе запроса прямого доступа к памяти ($\overline{\text{HOLD}}$), имеющегося во многих микропроцессорах, постоянно действует низкий уровень. С помощью логического пробника можно проверить и целостность печатных проводников; если, например, микросхема памяти не выбирается, следует проверить наличие импульсов на ее входе $\overline{CE}$ (разрешение работы кристалла) и проследить по печатному проводнику до того выхода дешифратора адреса, на котором формируется сигнал $\overline{CE}$. На печатных платах с высокой плотностью упаковки микросхем применяются очень узкие проводники, на которых могут появляться микроскопические разрывы. Ведя пробник по проводнику, можно обнаружить разрыв, незаметный для невооруженного глаза.

### 5.2. Логические пульсаторы

Логический пробник контролирует наличие уровней или импульсов только в одном узле схемы; в дополнение к логическому пробнику разработан логический пульсатор, который стимулирует узел, вынуждая его переходить из одного состояния в другое. Логические пульсаторы — это схемные стимулирующие приборы, предназначенные для введения («инжекции») в узел коротких и мощных импульсов, которые переводят узел из одного состояния в другое и возвращают в первоначальное состояние. Обычно пульсатор генерирует импульс тока значением до 0,75 А в течение 300 нс; благодаря малой длительности импульсов ИС не повреждается. Выходной каскад пульсатора тристабильный, поэтому при обычных условиях касание зондом узла в схеме не влияет на его поведение. Подача одиночного импульса в проверяемый узел осуществляется нажатием кнопки, находящейся на корпусе пульсатора. Зонд пульсатора оснащен индикатором, который вспыхивает

синхронно с выходным сигналом. Для срабатывания индикатора производится расширение выходного импульса, что делает вспышку заметной для пользователя.

Некоторые серийные пульсаторы, например, модели 546A фирмы Hewlett-Packard допускают настройку для работы с последовательностью импульсов. Пульсатор модели 546A может работать в шести режимах, устанавливаемых с помощью одного переключателя двойного действия. При нажатии переключателя формируется одиночный выходной импульс, а движение его вперед фиксирует режим. Таким образом, с помощью переключателя вручную можно зафиксировать в схеме управления пульсатора выбранный режим. Имеются следующие режимы работы пульсатора:

одиночный импульс;

непрерывная последовательность импульсов с частотой 100 Гц;

пачка (пакет) из 100 импульсов;

непрерывная последовательность импульсов с частотой 10 Гц;

пачка из 10 импульсов;

непрерывная последовательность импульсов с частотой 1 Гц.

Частота вспышек индикатора составляет 10 Гц при выходной частоте 100 Гц и 5 Гц при выходной частоте 10 Гц.

Схема управления в пульсаторе модели 546A запоминает число нажатий кнопки для выбора режима. Если, например, требуется пачка из 100 импульсов, необходимо нажать кнопку режима 2 раза, а затем продвинуть ее вперед. При каждом нажатии кнопки генерируется одиночный выходной импульс; эта информация запоминается в схеме управления, и поэтому в первой пачке будет 98 импульсов, а всего получается 100 импульсов. Каждая пачка из 100 импульсов длится одну секунду, а затем пульсатор ожидает одну секунду перед выдачей следующей пачки из 100 импульсов. Освобождение кнопки из заблокированного положения прекращает действие режима, но в пакетном режиме пульсатор заканчивает формирование текущей последовательности, а затем переводит выход в высокоимпедансное состояние.

Полные выходные сопротивления логических схем

достаточно велики, чтобы пульсатор мог мгновенно изменить состояние узла. Конечно, бессмыслению пытаться изменить состояние шины питания логическим пульсатором из-за малого выходного сопротивления блока питания.

### 5.2.1 Использование логического пульсатора

С помощью логического пульсатора в узлы схемы вводятся одиночные импульсы или последовательности импульсов, и для этого не нужно отключать микросхемы.

На рис. 5.7 показана типичная схема, применяемая в микрокомпьютерах для получения звукового сигнала. Усилителем мощности служит ТТЛ-элемент либо с тристабильным выходом, либо с открытым коллектором. ТТЛ-схемы могут отводить от нагрузки больший ток, чем отдавать, поэтому для достижения большей выходной мощности динамика или пьезоэлектрического звукового сигнализатора один его вывод подключается к источнику питания 5 В, а второй последовательно с резистором 100 Ом заземляется через выход ТТЛ-элемента. Когда выход ТТЛ-элемента находится в состоянии логической 1, ток в цепи отсутствует. При подаче на вход ТТЛ-элемента импульсов с определенной частотой динамик издает звуковой сигнал. Предположим, что из-за отказа звуковой сигнал не формируется, когда компьютер подает импульсы на вход ТТЛ-элемента, и динамик можно проверить, если коснуться зондом пульсатора входа ТТЛ-элемента и задать режим пачки из 100 импульсов или режим непрерывной последовательности импульсов с частотой 100 Гц. Если схема исправна, динамик издает звуковой тон с частотой 100 Гц. Когда же тест не проходит, то зонд пульсатора следует перенести на выход ТТЛ-элемента и повторить тест. Если тест вновь не проходит, пульсатором следует коснуться непосредственно динамика и, если его сопротивление не

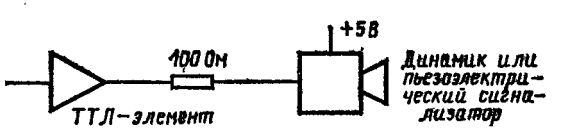



Рис. 5.7. Схема «звонка», часто применяемая в микрокомпьютерах

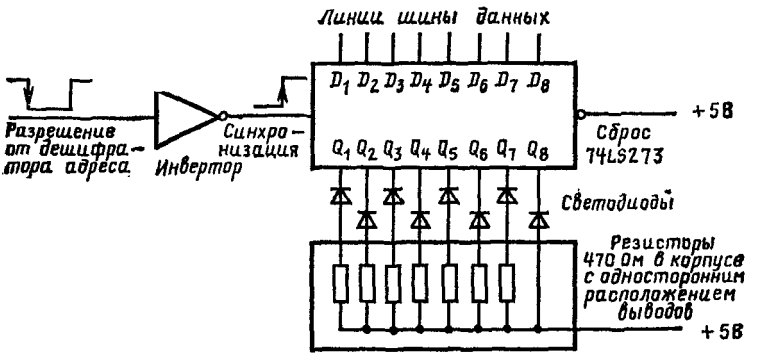



**Рис. 5.8.** Индикация выходного порта с помощью регистра-защелки 74LS273

слишком мало, чтобы препятствовать работе пульсатора, должен быть слышен звуковой тон. Таким образом, с помощью пульсатора можно проверить все компоненты этой простой схемы.

Индикаторный выходной порт, показанный на рис. 5.8, часто применяется в микроконтроллерах как простое средство отображения информации о состоянии отдельных узлов и шин. Информация с шины данных фиксируется в 8 D-триггерах, когда сигнал на входе синхронизации переходит в состояние логической 1. Обычно используемые в микропроцессорных системах сигналы выбора микросхем имеют низкий активный уровень, поэтому сигнал от дешифратора адреса до подачи на вход синхронизации микросхемы 74LS273 инвертируется.

Работу выходного порта можно проверить с помощью логического пульсатора, касаясь входа или выхода инвертора. При этом на D-триггеры воздействует активный фронт синхронизации, а информация из шины данных будет определять состояния светодиодных индикаторов. Если, например, линия шины данных, подключенная на вход $D_1$, при подаче импульса находится в состоянии логического 0, светодиод, подключенный к выходу $Q_1$, будет светиться. Сигналы на шине данных обычно изменяются столь быстро, что при подаче на вход синхронизации нескольких импульсов все светодиодные индикаторы будут включены и выключены одновременно. Если светодиод в течение всего времени пода-

чи синхроимпульсов остается выключенным, то следует предположить наличие одного из нескольких отказов, например неисправна линия шины данных, отказ в микросхеме 74LS273, отказ в светодиоде или в резисторе. Касание логическим пульсатором выхода микросхемы 74LS273 оказывается бесполезным, так как реакция светодиода на выходной импульс пульсатора будет незаметной пользователю. Однако подсоединение логического пробника к подозреваемому выходу при синхронизации порта решает эту проблему благодаря встроенной в логический пробник схеме расширения импульсов. Если к выходам порта индикаторы не подключены, для поочередной проверки каждого выхода следует применить логический пробник, синхронизируя порт несколько раз и фиксируя изменения состояния на каждом выходе.

Логический пульсатор модели 546А может генерировать точные пачки из 100, 10 или 1 импульса, что удобно при необходимости получения определенного числа импульсов для инициализации счетчика, регистра сдвига или другого конечного автомата.

Пусть имеется 12-битный асинхронный счетчик, реализованный на 4-битных ТТЛ-счетчиках 7493А, как показано на рис. 5.9. Требуется инициализировать счетчик на двоичный набор 1000 1000 1000, равный десятичному числу 2184. Пульсатор подсоединяется на вход счетчика и настраивается на получение пачки из 100 импульсов. После этого оператор отсчитывает 21 пачку и убирает пульсатор. Затем пульсатор вновь подсоединяется к входу и настраивается на формирование пачки из 10 импульсов; после этого оператор отсчитывает 8 пачек и удаляет пульсатор. Наконец, пульсатор еще раз подсоединяется к входу счетчика, и оператор вручную вводит 4 импульса, нажимая кнопку 4 раза. Таким образом,

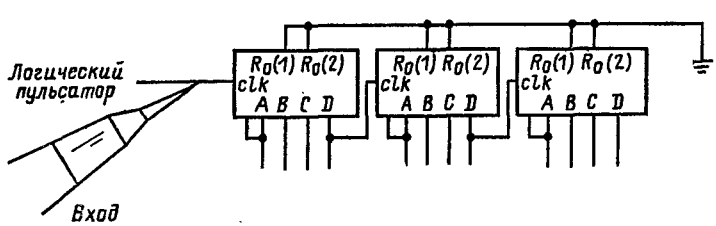



Рис. 5.9. 12-битный асинхронный счетчик

в счетчик поданы 2184 импульса, и он оказывается установленным на требуемое значение.

### 5.3. Тестирование «стимул—реакция» с помощью пульсатора и пробника

Общий метод контроля логических схем, при котором применяются логический пробник и логический пульсатор, называется тестированием «стимул — реакция». С помощью пульсатора в узел вводится стимулирующее воздействие, а получающаяся реакция логической схемы прослеживается логическим пробником. Зная проверяемую схему, исследователь может проследить логическим пробником тракт распространения сигнала.

По-видимому, простейшее и наиболее частое применение тестирования «стимул — реакция» заключается в проверке целостности соединения выхода одного элемента со входом другого элемента.

Для проверки схемы, приведенной на рис. 5.10, на вход элемента с уровнем логического 0 подаются сигналы от пульсатора и пробником проверяется выход $G_1$, чтобы убедиться в правильной работе элемента. Пульсатор можно оставить на входе элемента или перенести на его выход и коснуться пробником входа элемента. $G_2$. Если в соединяющей элементы линии есть разрыв, пробник показывает «плохой» логический уровень и не реагирует ни на какие стимулы пульсатора.

Нужно отчетливо понимать, что инициируемые пульсатором воздействия при наличии целостности соединений распространяются по сигнальному тракту, а это может привести к неожиданным или нежелательным реакциям системы. Обратимся, например, к схеме на рис. 5.10. Формирование импульсов в подозреваемой соединительной линии вызывает подачу сигналов синхро-

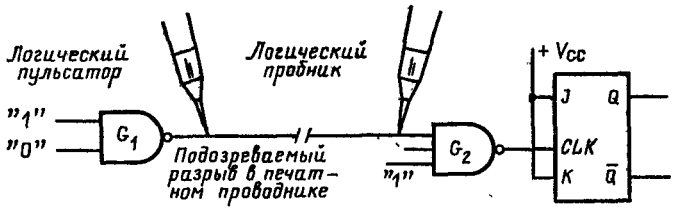



Рис. 5.10. Проверка целостности печатного проводника

низации с выхода элемента $G_2$ на $JK$-триггер, и он будет изменять свое состояние, если линия невредима. Изменение состояния триггера может вызвать запускающие действия в тракте распространения сигнала, что в конце концов вызовет изменение входов $G_1$, а также состояния проверяемой линии. Это может привести к путанице, если при первой проверке линия находилась в состоянии логической 1, а после подачи импульсов внезапно переходит в состояние логического 0. Для интерпретации такого изменения исследователь должен знать всю схему и ее возможное поведение, если он хочет получить из реакций схемы осмысленные результаты.

Хотя описанное тестирование «стимул — реакция» широко применяется для проверки логических устройств, реакции проще интерпретировать в комбинационных схемах, чем в конечных автоматах, так как поведение комбинационных схем не зависит от времени.

### 5.4. Индикатор тока

Индикатор (буквально «прослеживатель») тока фиксирует наличие изменяющегося тока в печатном проводнике с помощью индуктивной приемной катушки, находящейся в его зонде. Английский физик М. Фарадей открыл закон электромагнитной индукции 29 августа 1831 г., а годом позднее совершенно независимо это же явление открыл американский физик Д. Генри. Сформулированный Д. Генри закон гласит, что при изменении тока в индуктивной цепи наводится электродвижущая сила (ЭДС). Выражение, связывающее наводимую ЭДС с изменением тока, имеет вид $E=-Ldi/dt$, где $E$ — наводимая ЭДС; $L$ — индуктивность цепи; $di/dt$ — скорость изменения тока. Отрицательный знак объясняется законом Г. Ленца, который гласит, что наводимая ЭДС имеет направление, противодействующее изменению вызывающего ее тока. М. Фарадей продемонстрировал, что, если две цепи связаны магнитопроводом, изменение тока в одной цепи вызывает наведение ЭДС в другой. Изменяющийся ток в первичной цепи вызывает изменение магнитного потока в магнитопроводе, которое наводит ЭДС во вторичной цепи.

Закон Фарадея гласит, что если к индуктивной цепи обмотки на железном кольце внезапно приложить ЭДС,

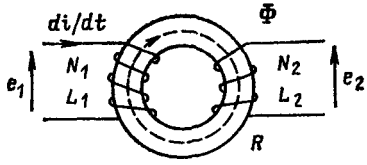



Рис. 5.11. Электромагнитная индукция

то она будет связана с изменением магнитного потока в железном кольце соотношением

$$e_1 = -N_1 d\Phi/dt,$$

где $N_1$ — число витков в первичной обмотке, а $\Phi$ — магнитный поток. Объединение выражений, описывающих законы Генри и Фарадея, дает

$$e_1 = -N_1 d\Phi/dt = -L_1 \frac{di}{dt},$$

откуда получаем

$$\frac{d\Phi}{dt} = \frac{L_1}{N_1} \frac{di}{dt}.$$

Это же самое изменение потока вызывает наведение ЭДС во вторичной обмотке (рис. 5.11), что описывается выражением

$$e_2 = -N_2 \frac{d\Phi}{dt}.$$

Следовательно, вторичная наводимая ЭДС

$$e_2 = -\frac{N_2}{N_1} L_1 \frac{di}{dt}.$$

Если первичная обмотка является частью печатного проводника, она, по существу, представляет собой одновитковую обмотку, т.е. $N_1 = 1$. Поэтому наводимая во вторичной обмотке ЭДС

$$e_2 = -N_2 L_1 \frac{di}{dt}.$$

Поскольку $N_2$ является константой, наводимая ЭДС $e_2$ прямо пропорциональна скорости изменения тока, протекающего в печатном проводнике.

Ток, протекающий в печатном проводнике, образует магнитное поле, и при изменении тока магнитный поток тоже изменяется. Такое изменение потока используется в индикаторе тока для наведения ЭДС в катушке, смонтированной в зонде индикатора. Конструкция катушки оказывает решающее влияние на индикацию, так как она должна быть по возможности небольшой, чтобы на

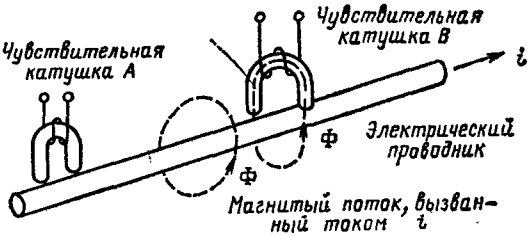



**Рис. 5.12.** Структура магнитного потока около круглого проводника

прослеживание изменяющегося тока в одном печатном проводнике не оказывали заметного влияния изменяющиеся токи в соседних проводниках.

Типичная поисковая катушка состоит из подковообразного ферритового магнитопровода с намотанной на него катушкой (рис. 5.12). Печатный проводник на плате можно считать электрическим проводником, около которого при протекании тока образуется круговой магнитный поток. Катушка *В* ориентирована таким образом, что вызванный током магнитный поток заходит в ее магнитную цепь. Если ток *i* изменяется, поток также изменяется, и в обмотке катушки *В* наводится ЭДС. Однако катушка *А* ориентирована так, что вызванный током *i* магнитный поток не заходит в ее магнитную цепь, поэтому изменение тока не наводит ЭДС в катушке *А*. Следовательно, катушку в зонде индикатора тока необходимо ориентировать в соответствии с направленнем, показанным на рис. 5.12 для катушки *В*, иначе в ней либо совсем не наводится ЭДС, лнбо наводится очень слабая ЭДС, и прибор не зафнксирует изменений тока. Обычно на зонде делается отметка, которую для получения максимально наводимой ЭДС необходимо ориентнровать по направлению проверяемого печатного проводника.

Индикатор обнаруживает только изменение тока, а при протекании в проводнике постоянного тока он совсем не дает показаний. Цифровые схемы являются импульсными схемами, в которых сигналы непрерывно переходят из одного состояния в другое, что, в свою очередь, заставляет изменяться токи, протекающие в электрических соединениях. Реальные значения токов могут быть небольшими, но временной интервал, в тече-

ние которого происходит изменение состояния, обычно очень мал, что обеспечивает измеримые значения *di/dt*. Индикатор тока, например, модели 547A фирмы Hewlett-Packard воспринимает изменения тока в диапазоне от 1 мА до 1 А. Он имеет встроенную регулировку чувствительности, которая позволяет установить интенсивность свечения индикаторной лампы. На практике пользователь регулирует индикатор тока таким образом, чтобы по интенсивности свечения лампы получить информацию об относительных значениях токов, протекающих в различных печатных проводниках. Для успешного применения индикатора тока необходим больший опыт, чем при использовании логического пробника или логического пульсатора, так как нужно уметь регулировать чувствительность и правильно ориентировать индикатор тока вдоль прослеживаемого проводника. Высокие плотности размещения микросхем и проводников на современных печатных платах часто затрудняют размещение зонда на проводнике, поэтому индикатор может давать информацию не только о прослеживаемом проводнике, но и о соседних проводниках.

Возможности индикатора тока в поиске неисправностей менее очевидны, чем возможности логического пробника или логического пульсатора, так как он предназначен для прослеживания тока, а не привычных уровней напряжения.

#### 5.4.1. Использование индикатора тока

Для правильной работы индикатора тока его зонд должен быть установлен перпендикулярно контролируемому проводнику и ориентирован по его направлению. Если проводник поворачивает, например, под прямыми углами, необходимо соответственно поворачивать и индикатор тока.

Зонд индикатора тока электрически изолирован, поэтому им можно касаться проводника. Когда индикатор тока помещается на одну из линий шины адреса работающей вычислительной системы, его чувствительность регулируется так, чтобы интенсивность свечения лампы составляла примерно половину максимальной. Если прибор был ориентирован правильно, то при вращении зонда интенсивность свечения лампы будет изменяться. При касании шины питания интенсивность свечения уве-

личивается. На практике при прослеживании тока в проводниках, не являющихся шинами питания, проще всего отрегулировать чувствительность так, чтобы лампа еле светилась, — это позволит прибору обнаруживать небольшие изменения тока. Обычно индикатор тока имеет схему расширения импульсов, которая дает возможность оператору наблюдать частые и быстрые изменения уровней тока.

### 5.4.2. Тестирование «стимул — реакция» с помощью логнческого пульсатора и индикатора тока

В общем, индикатор тока обнаруживает наличие импульсов в схеме хуже логического пробника, так как им труднее пользоваться. Однако для некоторых видов отказов эквивалентной замены индикатору тока нет и применение его экономит много времени. Во многих ситуациях, например при коротком замыкании на землю внутри ИС, уровень протекающего тока оказывается статическим, и при помощи одного индикатора его обнаружить невозможно. Для решения этой проблемы индикатор часто применяется вместе с логическим пульсатором, который стимулирует неисправную линию, а вызываемые пульсатором изменения тока обнаруживаются индикатором.

При использовании индикатора тока вместе с пульсатором для нахождения отказа в линии данных рекомендуется настроить его по выходу пульсатора в соответствии с рис. 5.13. Чувствительность индикатора тока регулируется так, чтобы получить относительно слабое свечение примерно посередине его диапазона яркости, при этом вводимый ток имеет значения в диапазоне 10—100 мА. Примерно такой уровень тока ожидается от выходного ТТЛ- или КМОП-элемента, управляющего линией в состоянии логического 0, к которой подключены входы нескольких элементов. Если, например, есть

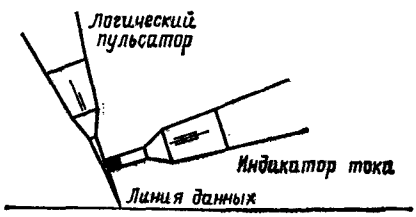



**Рис. 5.13.** Настройка индикатора тока для применения вместе с пульсатором

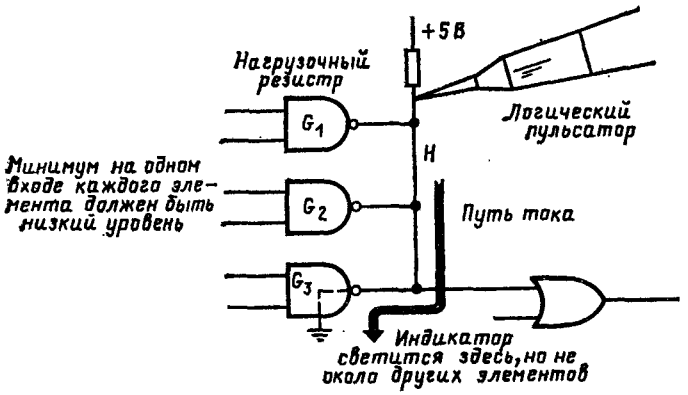



**Рис. 5.14.** Отказ узла монтажного И

короткое замыкание линии данных на землю, на выходе пульсатора действует ток около 1 А, и при любой настройке чувствительности лампа индикатора тока светится очень ярко. Опыт работы с индикатором тока позволяет делать обоснованные предположения о значении тока и его источнике.

**5.4.2.1. Узлы с монтажным И.** Одной из наиболее трудных проблем, встречающихся при поиске неисправностей в цифровых ИС, является «зависший» узел с монтажным И (рис. 5.14).

Выходы типа открытого коллектора элементов $G_1$—$G_3$ (НЕ-И) объединены и подключены к $V_{cc}$ через нагрузочный резистор. Данная схема называется монтажным И, так как реализуемая в ней булева функция имеет вид $H = \overline{(A \cdot B)} \cdot \overline{(C \cdot D)} \cdot \overline{(E \cdot F)}$.

Предположим, что один из выходных транзисторов элементов $G_1$, $G_2$ или $G_3$ постоянно закорочен на землю. Тогда с помощью логического пробника можно убедиться, что узел всегда имеет низкий уровень независимо от состояний входов $A$—$F$, но обнаружить отказавший элемент логическим пробником нельзя. Отказ может быть в любом из элементов $G_1$—$G_3$, и, кроме того, это может быть короткое замыкание в элементе $G_4$ или замыкание на землю самой линии.

Предположим, что причиной замыкания является выход элемента $G_3$. Тогда в узел можно подать выходные сигналы логического пульсатора, проследить с помощью

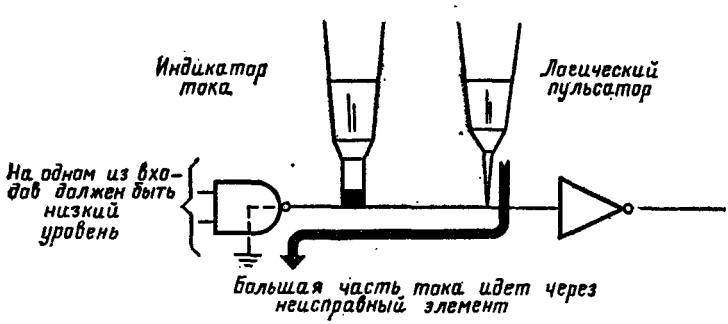



**Рис. 5.15.** Отказ элемент-элемент

индикатора тока путь тока на землю и выявить неисправный элемент $G_3$ среди нескольких; альтернативой является поочередное отсоединение выходов каждого из элементов и проверка исчезновения отказа, на что, разумеется, уйдет много времени.

**5.4.2.2. Отказы элемент — элемент.** Когда между двумя элементами существует низкоимпедансный отказ, с помощью индикатора тока и логического пульсатора можно быстро найти неисправный элемент.

На рис. 5.15 показан элемент, выход которого внутри корпуса ИС закорочен на землю. Такой отказ можно локализовать, пользуясь пульсатором и индикатором тока. Необходимо поместить пульсатор примерно посередине между двумя элементами и настроить его на непрерывную последовательность импульсов с частотой 100 Гц. Затем нужно коснуться зондом индикатора тока зонда пульсатора и отрегулировать чувствительность на слабое свечение лампы. Индикатор тока следует поднести сначала к элементу $G_1$, а затем к элементу $G_2$, продолжая стимулировать проводник пульсатором. Индикатор тока будет светиться только у элемента $G_1$, так как он неисправен и через него протекает почти весь ток. Если отказ вызван коротким замыканием входа элемента $G_2$, индикатор будет светиться только у этого элемента.

**5.4.2.3. Замыкание припоем и проверка кабелей.** При проверке печатных проводников на короткие замыкания, вызванные припоем или плохим травлением, пульсатор помещают в начало линии и прослеживают путь основного тока. Рассматриваемый тип отказа часто возникает при производстве печатных плат, когда замыкание двух

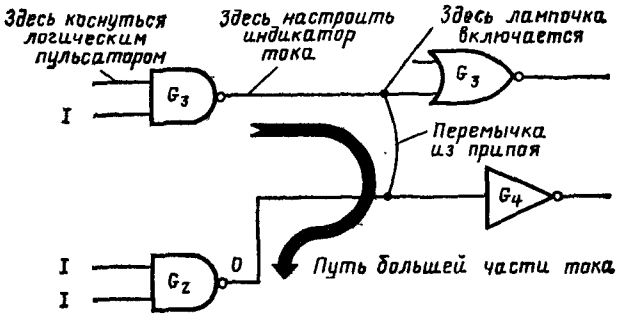



**Рис. 5.16.** Перемычка из припоя на печатной плате

линий вызывается ручной пайкой или пайкой волной. Так же иногда отыскивают замыкания в кабелях.

Пример на рис. 5.16 показывает неправильный путь тока, появившегося из-за мостика припоя между двумя проводниками. Когда индикатор тока перемещается по проводнику от выхода элемента $G_1$ к элементу $G_3$, его лампа светится до тех пор, пока не будет пройден мостик припоя. Затем лампа гаснет, показывая, что ток течет по какому-то другому пути, чем путь по проводнику от элемента $G_1$ к элементу $G_3$. Визуальный осмотр того места, где погасла лампа, позволяет обнаружить отказ.

**5.4.2.4. Входы нескольких элементов.** Общий вид логической структуры, в которой выход одного элемента подключен на входы нескольких элементов, представлен на рис. 5.17. В большинстве логических элементов на всех их входах имеются защитные диоды, которые либо защищают прибор от статического заряда в буферированных КМОП-схемах, либо ускоряют реакцию элемента (диоды Шоттки в ТТЛ-схемах). Эти диоды могут быть «пробиты», что приведет к появлению низкоимпедансного тракта на землю. К одной линии могут быть подключены входы нескольких элементов, и проблема заключается в определении неисправного входа.

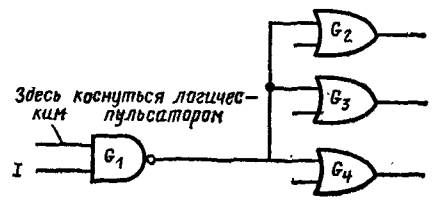



**Рис. 5.17.** Входы нескольких элементов

На вход элемента $G_1$ подается выход логического пульсатора, а индикатор тока регулируется на выходе $G_1$ так, чтобы его лампа слабо светилась. Затем с помощью индикатора тока проверяется наличие тока в каждом из элементов $G_2$—$G_4$. Лампа будет светиться только на неисправном входе. Если при регулировке индикатора тока на выходе элемента $G_1$ его лампа не светится, это служит признаком того, что отказ скрыт в элементе $G_1$. Его можно проверить с помощью пульсатора и индикатора тока, как было показано в п. 5.4.2.2.

**5.4.2.5. Короткие замыкания в цепях питания.** Одним из наиболее труднолокализуемых отказов является короткое замыкание в шинах питания на печатной плате. Напряжение питания 5 В, принятое в большинстве логических семейств, подводится почти к каждой ИС на печатной плате; к шинам питания подключено также множество керамических и танталовых конденсаторов. На схемной плате с 20 микросхемами находится 10—15 развязывающих конденсаторов. В ИС может возникнуть короткое замыкание между питанием и землей без видимых проявлений. Но обычно при таком отказе в центре корпуса DIP наблюдается заметное почернение. Впрочем, подобный отказ возникает относительно редко. Однако короткое замыкание в развязывающих конденсаторах встречается гораздо чаще, а так как все конденсаторы включены параллельно, для нахождения дефектного конденсатора приходится по очереди отключать каждый из них. На печатных платах с высокой плотностью упаковки элементов такая операция оказывается сложной и длительной.

Рассматриваемый отказ можно локализовать путем отключения питания от печатной платы, возбуждения шины питания при помощи логического пульсатора и прослеживания пути протекания тока индикатором тока. Для питания пульсатора и индикатора тока необходим отдельный источник (обычно пользуются блоком, отключенным от платы), а подключение к земле нужно оставить на плате. Пульсатор настраивается на режим непрерывных импульсов с частотой 100 Гц, а индикатор тока регулируется в соответствии с рис. 5.13. Затем индикатор тока перемещается по шине питания до достижения места, где его лампа выключается. Возможно, что здесь шина питания разделяется, и индикатор нужно оставить на том проводнике, в котором течет ток. В конце

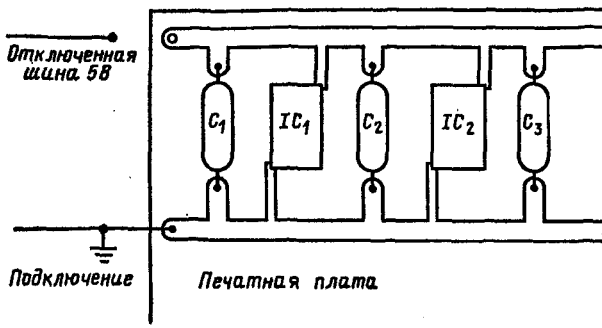



**Рис. 5.18.** Короткое замыкание в цепи питания

концов индикатор окажется в такой точке, где разветвления нет, а его лампа гаснет. В обнаруженной области находится неисправный элемент, которым может быть конденсатор или ИС. Если ИС вставлена в панельку, прежде всего микросхему нужно вынуть и посмотреть, не исчез ли отказ. Если отказ сохраняется, то подозрение падает на развязывающий конденсатор, находящийся в просматриваемой области.

На рис. 5.18 предполагается, что короткое замыкание имеется в конденсаторе $C_2$. Питание 5 В подается на пульсатор и индикатор тока от блока питания, который отключен от печатной платы. Зонд пульсатора касается контакта питания разъема печатной платы, и индикатор тока перемещается по шине питания до тех пор, пока его лампа не погаснет. В данном примере это произойдет сразу после конденсатора $C_2$, что указывает на то, что именно он является дефектным элементом.

### 5.5. Логические компараторы

Некоторые ИС можно проверить в системе, пользуясь прибором, который называется логическим компаратором (рис. 5.19). Компаратор содержит плату с микросхемой, аналогичной проверяемой, и имеет клипсу, с помощью которой он подключается к проверяемой микросхеме. В процессе работы компаратор воспринимает все входные сигналы проверяемой микросхемы и подает их на встроенную в него микросхему. Выходы обеих микросхем сравниваются с помощью элемента исключающего ИЛИ, и при их различии сформированный сигнал ошибки расширяется и подается на светодиод.

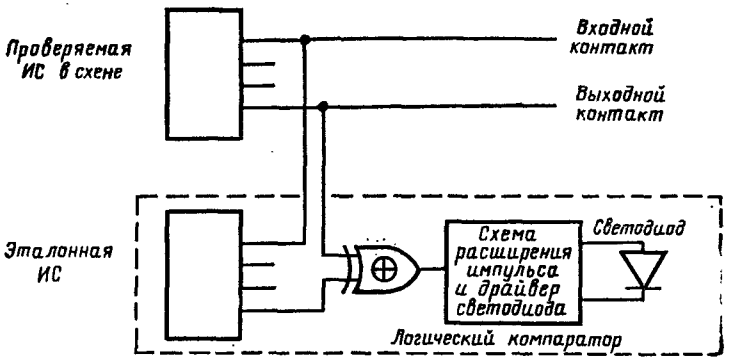



**Рис. 5.19.** Принцип действия логического компаратора

Компаратор освобождает исследователя от анализа длинных и сложных последовательностей сигналов, автоматически указывая, что сигнал имеет хороший «высокий» илн «низкий» уровни и что он соответствует таблице истинности проверяемой микросхемы.

Необходимо помнить, что логический компаратор нагружает проверяемую микросхему, и если она уже работает с максимальным коэффициентом разветвления по выходу, то это может привести к ее хаотичному поведению. Например, компаратор 10529А фирмы Hewlett-Packard создает нагрузку в 1,2 входной нагрузки на любой вход, к которому он подключается, и всего 0,1 ТТЛ-нагрузки на любой выход. В большинстве ситуаций эти дополнительные нагрузки не влияют на работу проверяемой схемы, но в схемах с аналоговыми компонентамн, например с одновибратором, дополнительная нагрузка может изменить временные характеристики.

Конечные автоматы также ставят проблемы перед компараторами, так как их выходы зависят от значений предшествующих входных сигналов, а эталонная микросхема компаратора может не иметь тех же самых наборов «прежних» входных сигналов. Следовательно, до выполнения сравнений проверяемую микросхему необходимо сбрасывать.

Значительное число различных микросхем в любом семействе логических элементов приводит к тому, что некоторые из них нельзя проверить с помощью компаратора; обычно к таким микросхемам относятся все

драйверы-дешифраторы индикаторных ламп, например ТТЛ-схемы 7447. Еще один недостаток логического компаратора — это необходимость иметь для каждой проверяемой микросхемы отдельную эталонную плату. В настоящее время существует семь различных ТТЛ-семейств элементов, содержащих сотни микросхем, и поэтому число эталонных плат оказывается чрезмерно большим.

### 5.6. Ограниченные возможности ручных инструментальных средств

Логические пробники и пульсаторы, индикаторы тока и логические компараторы продолжительное время доминировали в качестве инструментальных средств поиска неисправностей в цифровых схемах. Однако им свойственны ограничения в том смысле, что пульсатор может возбуждать одиовременно только ограниченное число узлов в системе, а логический пробник проверяет только один узел. Пробник может дать полезную информацию о статическом состоянии узла или показать наличие импульсов в цепи, однако он не может дать содержательной информации о последовательностях импульсов. Ручные инструментальные средства играют важную роль при анализе отказов в обычных логических схемах, но они почти бесполезны при анализе систем с шинной структурой, где информация обновляется последовательно во времени на большом числе линий одновременно. Чтобы разобраться в работе микропроцессорной системы, исследователю требуются приборы, которые фиксируют и индицируют в удобной форме информацию со многих линий и могут выделить нужную ему информацию. Очевидно, что простым инструментальным средствам такие функции недоступны, что привело к необходимости разработки аппаратуры, предназначенной для поиска неисправностей в сложных системах с шинной структурой.

## 6. Логические анализаторы

В микропроцессорной системе информация, относящаяся к одной операции, существует на многих линиях в течение короткого интервала времени. При каждом считывании команды из памяти микропроцессор вна-

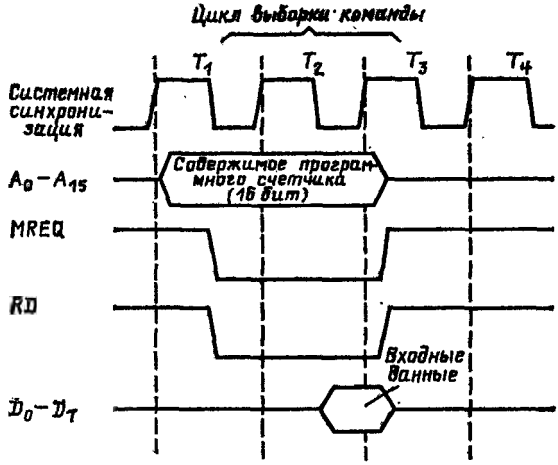



**Рис. 6.1.** Цикл выборки кода операции команды

чале помещает адрес кода операции на шину адреса. После этого он формирует управляющий сигнал запроса *памяти* (в системе с ВВ, отображенным на адресное пространство ВВ) и посылает управляющий сигнал считывания из выбранной микросхемы памяти. Выбранная микросхема памяти помещает код операции на шину данных, а ЦП воспринимает его и загружает в свой *регистр команды*. Такой цикл выборки команды необходимо выполнять всякий раз, когда код операции или операнд считываются из памяти, и он выполняется за три такта системной синхронизации.

По нарастающему фронту импульса синхронизации $T_1$ содержимое программного счетчика выдается на линии шины адреса, которая в типичном 8-битном микропроцессоре состоит из 16 линий. Половина такта синхронизации отводится на стабилизацию состояний на этих линиях, а затем выдаются управляющие сигналы запроса памяти и считывания. Эти сигналы показаны на рис. 6.1 как $\overline{MREQ}$ и $\overline{RD}$; обычно они являются сигналами низкого уровня. В различных микропроцессорах мнемоники данных сигналов могут быть различными. В системе с отображением ВВ на адресное пространство памяти управляющей линии запроса памяти может и не быть. Микропроцессор ожидает еще 1,5 периода синхронизации до опроса линий шины данных по нарастающему

фронту импульса синхронизации $T_3$. Это время отводится устройству памяти на то, чтобы дешифрировать адрес и поместить содержимое адресуемой ячейки на шину данных. Ожидается, что за это время информация на линиях шины данных стабилизируется. Если память не может отреагировать за отведенное время, она должна выдать в ЦП сигнал ожидания $\overline{WAIT}$, чтобы ЦП не считывал с шины неустановившиеся («плохие») данные. Сразу после приема данных с шины ЦП снимает управляющие сигналы $\overline{MREQ}$ и $\overline{RQ}$, а также адрес с шины адреса. Все микропроцессоры выполняют операции считывания из памяти так, как показано на рис. 6.1, хотя детали этого процесса могут различаться. Временная диаграмма цикла выборки команды приводится в техническом описании любого микропроцессора.

Изучение рис. 6.1 показывает, что вся информация, необходимая для декодирования адреса, типа выполняемой операции и относящихся к операции данных, доступна в стабильной форме только по нарастающему фронту импульса синхронизации $T_3$. Следовательно, если зафиксировать и запомнить состояния шин адреса, управления и данных в этот момент времени, то будет получена вся информация, необходимая для интерпретации выполняемой операции. Однако для достижения этого потребуется запомнить состояния 16 линий адреса, 8 линий данных, минимум 2 управляющих линий и синхронизировать момент восприятия данных с системной синхронизацией. Всего получается 27 линий, информацию с которых необходимо воспринимать и запоминать. Еще одна проблема связана с тем, что данная диаграмма может относиться только к выборке кода операции, а при считывании из памяти операнда адрес сохраняется на шине дольше и шина данных опрашивается по спадающему фронту импульса синхронизации $T_3$. К счастью, большинство микропроцессоров выдают какой-либо управляющий сигнал, который информирует другие компоненты системы о выборке именно кода операции, а не операнда.

Реальные формы сигналов, действующих на системных шинах, не играют существенной роли. Важны только их логические состояния в те моменты времени, когда они воспринимаются прибором и содержат необходимую информацию. Следовательно, приборы могут запоминать системные сигналы в виде «чистых» двоичных сигналов и хранить их в своей внутренней цифровой

памяти. Такие приборы, позволяющие воспринимать и запоминать для последующей индикации текущие состояния вычислительной системы, называются анализаторами логических состояний.

### 6.1. Анализаторы логических состояний

Анализаторы логических состояний воспринимают и запоминают информацию от цифровых систем. Они прошли эволюцию от сравнительно простых приборов, построенных на базе осциллографа, до сложных и гибких средств анализа.

Чтобы разобраться в возможностях современных анализаторов логических состояний, целесообразно рассмотреть историю их развития. Отправной точкой следует признать заслуженный осциллограф. Осциллограф с мультиплексором на входе может показать гораздо больше сигналов, чем обычный двухканальный прибор. Временна́я развертка является общей для всех входных сигналов, а ее пилообразный сигнал заставляет луч перемещаться по экрану слева направо. Аналогичная пила действует по оси *Y*, и в определенных ее точках один из нескольких сигналов опрашивается и индицируется как точка на экране. С помощью мультиплексирования можно обрабатывать до 8 входных сигналов, что позволяет индицировать на экране 8 различных сигналов от вычислительной системы. Для запуска развертки применяются сигналы синхронизации вычислительной системы или сигналы, сформированные из них.

Информацию, представленную на осциллографе, как показано на рис. 6.2, расшифровать и проанализировать невозможно из-за высокой скорости изменения индицируемых данных, представления их в виде набора прямо-

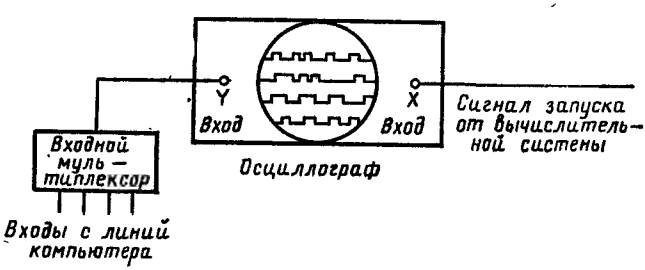



**Рис. 6.2.** Многоканальный осциллограф

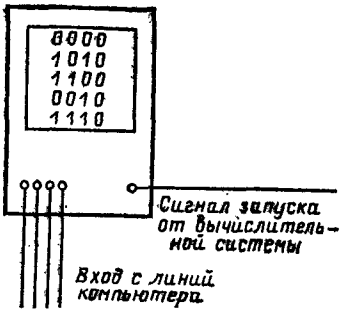



**Рис. 6.3. Многовходовый осциллограф с двоичной индикацией**

угольных сигналов и неопределенности точки запуска развертки. Один ход развертки со сканированием информации можно воспринять и запомнить с помощью запоминающего осциллографа, который решает первую проблему. Но все же остаются трудности в определении того, к какой операции компьютера относятся индицируемые данные и какое событие запустило индикатор. Индикацию можно сделать более понятной, если индицировать данные в виде двоичных единиц и нулей вместо прямоугольных сигналов, но тем не менее и здесь сохраняются сложные проблемы определения того, что представляют собой индицируемые данные.

Индикатор на рис. 6.3 более удобен в работе, так как способен отображать 16-ричные символы 0, *A*, *C*, *2* и *E*. Коды адресов и данных в 8-битной микропроцессорной системе обычно представляются как 16-ричные символы. Если необходимо индицировать более 4 бит, например адрес и соответствующее значение данных, для улучшения восприятия двоичный набор разделяется на группы по 4 бита (или группы по 3 бита, если применяется 8-ричная система счисления). Следующая запись может представлять собой индикацию значений адресов и данных в варианте системы, показанной на рис. 6.3:

```
0000 0000 0000 0000   0011 1110   0000 3E
0000 0000 0000 0001   0000 0110   0001 06
0000 0000 0000 0010   1101 0011   0002 D3
0000 0000 0000 0011   0000 1000   0003 08
```

Каждый 16-битный адрес представлен в виде 4 групп по 4 бита, а данные показаны как 2 группы по 4 бита. Справа приведены эквивалентные 16-ричные коды.

Информацию более не нужно индицировать как аналоговые сигналы, и обычный запоминающий осциллограф не требуется. Адреса и данные можно хранить во внутреннем ЗУПВ прибора. Содержимое ЗУПВ можно считывать без разрушения и индицировать на обычном

осциллографе в течение любого временно́го интервала.

При проверке микрокомпьютера исследователь заинтересован в просмотре вполне определенных фрагментов программы, а не каких-то событий, зафиксированных анализатором. Для этого требуется некоторая логика признаков, которая позволит прибору воспринимать информацию от проверяемой системы только при удовлетворении определенных условий. Обычно таким признаком является адрес, с которого начинается программная последовательность. Следовательно, логика признаков должна представлять собой набор логических компараторов, на которые подаются те условия, удовлетворение которых необходимо для того, чтобы анализатор начал воспринимать информацию от проверяемой системы. В первых образцах анализаторов логических состояний информация о признаках вводилась с помощью трехпозиционных переключателей, представляющих состояния «1», «0» и «безразлично».

Удовлетворяющая данным требованиям структурная схема анализатора представлена на рис. 6.4. Все линии шины адреса проверяемой системы подключены к 16-битному компаратору, а на другие 16 входов поданы сигналы от трехпозиционных переключателей. Аналогичным образом линии шины данных и еще 8 переключателей поданы на второй компаратор. Обычно предусматривается возможность подачи еще 8 сигналов в третий компаратор.

Когда адрес на шине совпадает с набором на адресных переключателях, на выходе $A=B$ верхнего компаратора появляется сигнал 1. Он разрешает работу второму компаратору, подключенному к линиям шины данных. Если код на шине данных также совпадает с набором на переключателях, то на выходе $A=B$ появляется сигнал 1, который, в свою очередь, разрешает работу третьему компаратору. Если в этот же момент времени состояния сигналов в 8 других линиях от проверяемой системы совпадают с состояниями, набранными на соответствующих переключателях, на выходе $A=B$ третьего компаратора появляется сигнал 1. Он разрешает работу совокупности вентилей, которые пропускают сигналы адреса, данных и другие сигналы в ЗУПВ с разрядностью слова 32 бита. Информацию от проверяемой системы чаще всего запоминают только по совпадению значений адреса. Предположим, например, что на переключателях ши-

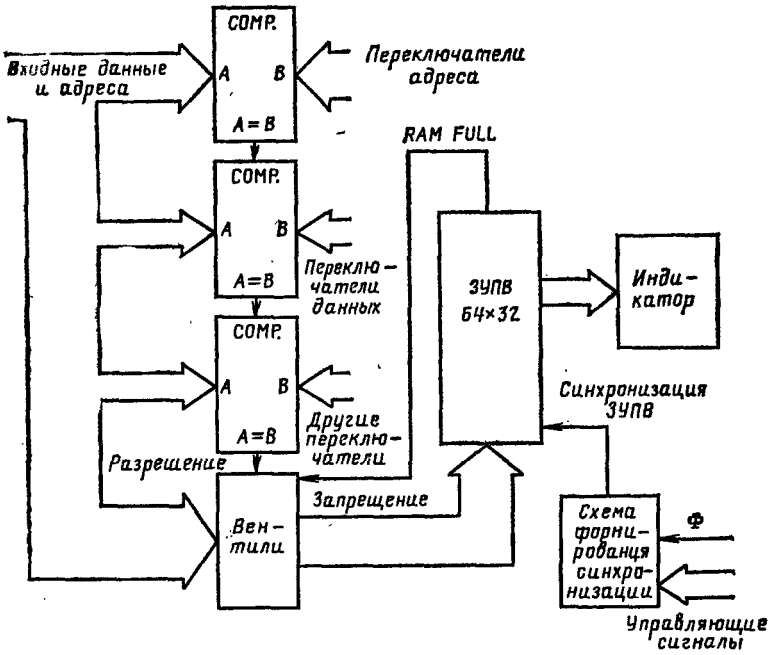



**Рис. 6.4.** Анализатор логических состояний

ны данных и переключателях, подсоединенных к компаратору управляющих сигналов, установлены «безразличные» состояния. Тогда при совпадении входного адреса с положениями переключателей выходной сигнал $A=B$ компаратора проходит через два других компаратора и разрешает работу входных вентилей ЗУПВ. Аналогичным образом информацию можно воспринимать, пользуясь заданием только значений данных.

Показанное ЗУПВ запоминает до 64 полных слов информации, состоящих из 16-битного адреса, 8-битного кода данных и состояний других 8 линий от системы. Следовательно, слова ЗУПВ должны иметь формат 32 бита. Когда в ЗУПВ зарегистрированы 64 системных события, оно формирует сигнал *RAM FULL*, который запрещает работу входных вентилей. После этого все 64 слова из ЗУПВ можно выводить на индикатор для анализа.

Оператору часто требуется просмотреть только определенные события на шине, например операции ВВ. Для

этого из сигналов системной синхронизации формируются сигналы синхронизации ЗУПВ анализатора, но они обусловливаются дополнительными управляющими сигналами. Например, запись в ЗУПВ анализатора может производиться только при наличии в системе сигнала запроса ВВ, и поэтому в ЗУПВ запоминаются только операции ВВ. Таким образом, пользователь может выбирать тип информации, воспринимаемой от системы, определяя не только информацию на шинах адреса и данных, но и типы запоминаемых событий.

Однако в системе, показанной на рис. 6.4, имеется один недостаток — она запоминает и индицирует события, возникающие только после появления запускающего слова, набранного на переключателях компараторов. Следовательно, от проверяемой системы воспринимаются только задержанные во времени события. При возникновении же неисправности причину ее часто можно определить, если зарегистрировать и проанализировать на индикаторе те события, которые привели к неисправности. Такая возможность, называемая предзапуском, позволяет оператору зарегистрировать и просмотреть как события до появления запускающего слова, набранного на переключателях, так и задержанную информацию. Схему, приведенную на рис. 6.4, следует модифицировать введением механизма предзапуска.

В модифицированном анализаторе, показанном на рис. 6.5, один набор информации подается в логические компараторы, как и раньше, от переключателей, но второй набор информации подается не с шин проверяемой системы, а из ячейки ЗУПВ. Записью информации в ЗУПВ по-прежнему управляет аналогичная схема признаков (квалификаторов) синхронизации, но теперь ЗУПВ работает как очередь FIFO («первый пришел — первый ушел»). Важное отличие заключается в том, что информация от проверяемой системы непрерывно подается в ЗУПВ до тех пор, пока запускающее слово на переключателях не совпадет с набором данных из той ячейки, которая подсоединена к логическим компараторам. При совпадении входные вентили закрываются, и содержимое ЗУПВ больше не изменяется. После этого данные из ЗУПВ подаются на индикатор, и их может проанализировать оператор.

Для логического сравнения выбираем любую из 64 ячеек ЗУПВ. Предположим, например, что для анали-

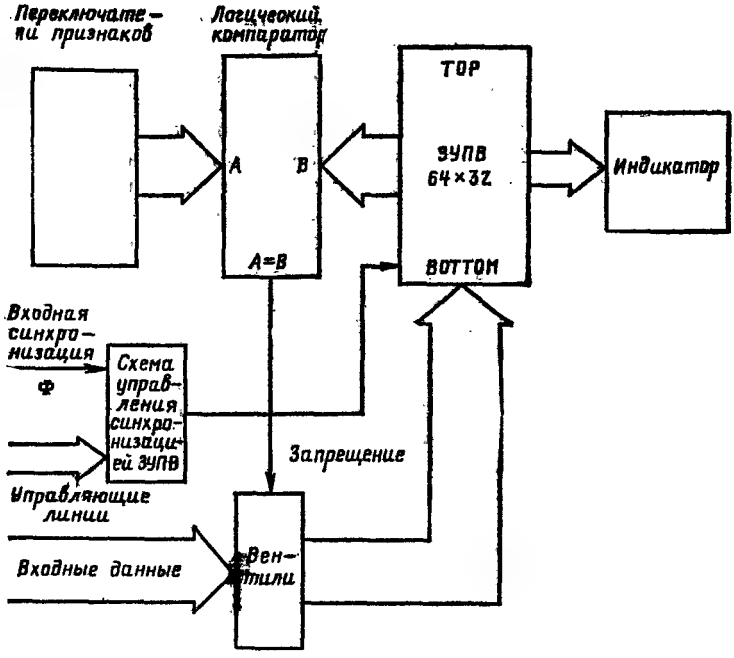



**Рис. 6.5.** Анализатор логических состояний с возможностью предзапуска

за выбрана верхняя ячейка ЗУПВ. Когда находящийся в этой ячейке двоичный код совпадает с набором на переключателях, входные вентили закрываются. При рассмотрении сверху вниз запомненные данные соответствуют событиям, которые возникли после запускающего слова (т. е. все они задержаны), и система работает аналогично системе, приведенной на рис. 6.4.

Когда же для анализа выбрана нижняя ячейка ЗУПВ при совпадении ее содержимого с набором на переключателях, то работа ЗУПВ запрещается и вся запомненная информация относится к событиям, которые предшествовали запускающему слову (их называют предзапусковыми событиями).

При выборе для анализа ячейки ЗУПВ между верхней и нижней ячейками ЗУПВ содержит само запускающее слово, несколько задержанных значений и несколько предзапусковых значений. Показанная на рис. 6.5 схема удовлетворяет необходимости регистрации и ин-

дикации событий, предшествующих некоторому запускающему событию, и позволяет индицировать до 63 предзапусковых событий в компьютере. Когда же нужно вернуться ранее этого предела, то на переключателях следует набрать код события, возникающего на 63 события раньше требуемого запускающего события, и тогда будут регистрироваться и индицироваться 63 события до запуска. Просто отмечая начальное значение индицируемой информации, пользователь может возвращаться от события блоками по 64 слова настолько, насколько ему необходимо.

Системы, представленные на рис. 6.4 и 6.5, дают общее представление об анализаторах логических состояний. На них не показано, каким образом компаратор обрабатывает «безразличные» состояния и как информация из ЗУПВ подается на индикатор для анализа пользователем. В большинстве анализаторов применяются логические схемы для обработки поступающих данных, чтобы обеспечить необходимое быстродействие, и микропроцессор, осуществляющий общее управлеиие.

Введение в анализатор логических состояний микрокомпьютера как общего управляющего элемента позволяет реализовать в приборе несколько удобных возможностей. На рис. 6.6 показана структурная схема основных компонентов типичного анализатора, в котором сохранены быстродействующие логические элементы для восприятия данных от *проверяемой системы* и микрокомпьютер, осуществляющий общее управление.

Для снижения электрической нагрузки проверяемой системы информация, воспринимаемая с шин проверяемой системы, подается в буферы анализатора. Выходы буферов подаются в логические компараторы, которые фиксируют заданный запускающий набор, и в ЗУПВ, выполненное по схеме FIFO. Наборы переключателей, которые применялись в прежних анализаторах, теперь заменены на выходные порты микрокомпьютера. Двоичные коды адреса, данных и необязательных входов, образующие запускающее слово, вводятся в микрокомпьютер с клавиатуры; аналогичным образом вводится число предзапусковых событий, необходимых до запускающего слова. ЗУПВ может хранить только 64 слова информации, и по введенному предзапусковому числу микрокомпьютер может вычислить число задержанных событий, которое должно произойти до заполнения ЗУПВ.

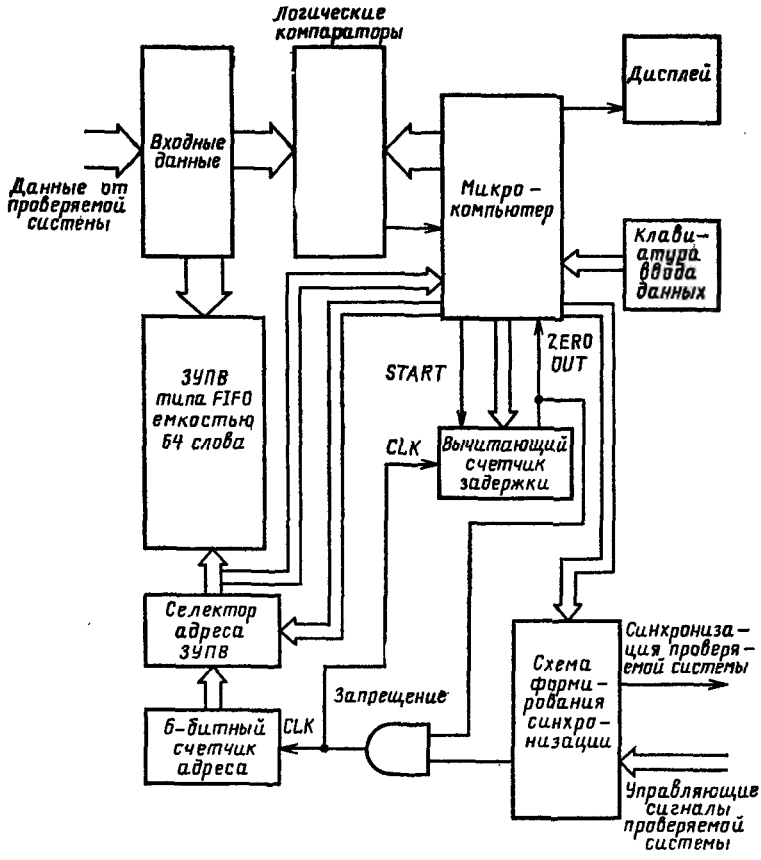



**Рис. 6.6.** Анализатор логических состояний со встроенным микрокомпьютером

Если, например, в систему введено предзапусковое значение 10, компьютер вычисляет, что в ЗУПВ до его заполнения должно находиться 53 задержанных события. Это значение загружается из выходного порта микрокомпьютера в вычитающий счетчик задержки.

В компьютер с клавиатуры вводится также тип системных событий, которые должны запоминаться в ЗУПВ. На основе этой информации осуществляется программирование схемы признаков синхронизации, которая управляет адресацией ЗУПВ со структурой FIFO.

После ввода с клавиатуры всей необходимой информации с клавиатуры же отдается приказ о запуске анализатора. Селектор данных, который определяет источник адресов ЗУПВ, программируется на подключение к ЗУПВ 6-битного счетчика адреса.

Когда информация от проверяемой системы совпадает с кодами, поданными в логические компараторы от компьютера, сигнал от компараторов сообщает компьютеру о том, что обнаружено запускающее событие. После этого компьютер считывает из 6-битного счетчика адрес, по которому запомнено запускающее событие, и запускает вычитающий счетчик задержки. Когда прошло вычисленное число задержанных событий, сигнал *ZERO OUT* от вычитающего счетчика запрещает дальнейшее запоминание, отключая синхронизацию 6-битного счетчика адреса. Этот же сигнал подается в компьютер, показывая, что теперь все необходимые события запомнены в ЗУПВ.

По существу, ЗУПВ действует как круговая память— когда данные записываются в ячейку с минимальным адресом, счетчик адреса переходит на максимальное значение, и следующий элемент информации запоминается в ячейке с максимальным адресом. Для слежения за событиями в ЗУПВ компьютер использует адрес, считанный из селектора адреса ЗУПВ, когда логические компараторы фиксируют появление запускающего слова (рис. 6.7).

Общее число ячеек в ЗУПВ известно (в нашем примере оно равно 64). С известными адресом запускающего слова в ЗУПВ и числом введенных с клавиатуры предзапусковых событий компьютер имеет всю необходимую информацию о способе размещения в ЗУПВ предзапусковых, запускающего и задержанных событий.

Для обеспечения быстродействия адрес ЗУПВ, соответствующий хранению запускающего слова, фиксируется выходным сигналом логических компараторов. При этом устраняется временная задержка на считывание адреса ЗУПВ микрокомпьютером.

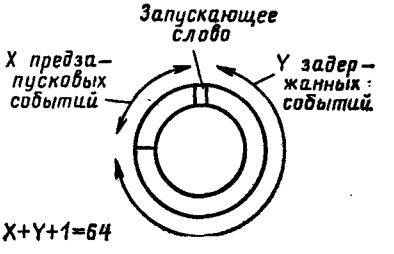



**Рис. 6.7.** Представление событий в круговом ЗУПВ

После регистрации данных в ЗУПВ микрокомпьютер может обращаться к ним с меньшей скоростью, управляя селектором адреса ЗУПВ и загружая адрес ЗУПВ из выходного порта. Анализатор переходит в режим индикации, в котором для управления индикацией хранимых данных (обычно на дисплее) используется клавиатура.

Появление в системе микрокомпьютера позволяет реализовать многочисленные форматы индикации. Вместо двоичных наборов, которые индицировались в прежних анализаторах, значения адресов и данных показываются в виде 16-ричных чисел. Система становится более «дружественной к пользователю», представляя хранимую информацию в более удобной форме.

Компьютер можно запрограммировать на «дисассемблирование» 16-ричных кодов операций конкретного микропроцессора и представлять на экране дисплея мнемокоды операций вместо их 16-ричных эквивалентов. Это значительно улучшает восприятие программного сегмента, так как упрощает и ускоряет отладку программы.

Объединение ввода данных с клавиатуры, микрокомпьютера и дисплея привело к появлению анализаторов логических состояний, спроектированных с учетом требований эргономики. С помощью дисплея можно помогать пользователю при настройке параметров, регистрации данных, индицируя на экране таблицу отмеченных элементов информации, необходимой для программирования компараторов и схемы признаков синхронизации. Возможно также информировать пользователя о вводе недействительного параметра. В режиме индикации на экране обычно нельзя показать всю хранимую информацию, поэтому для просмотра всех данных пользуются клавишами «рулона» (или «прокрутки»). Запускающее слово часто выделяется с помощью инверсного изображения, а в режиме «рулона» не допускается переход от последнего задержанного события к первому предзапусковому событию. Следовательно, оператор рассматривает информацию только в той последовательности, в какой она возникает в системе, и не испытывает замешательства из-за переходов от события, которое возникло после запускающего слова, к событию, которое возникло до него.

Фрагмент типичного дисассемблерного формата в

анализаторе логических состояний, запрограммированном на анализ микропроцессора Z80 фирмы Zilog, имеет следующий вид:

| АДРЕС | КОП/ДАННЫЕ | | ВНЕШНИЕ |
|---|---|---|---|
| 0D00 | LD | A, 06 | 1111 0000 |
| 0D02 | OUT | (08) | 1111 0000 |
| 0808 | 06 | OUTPUT | 1111 0001 |
| 0D04 | CALL | 0F06 | 1111 0011 |
| 0FFF | 0D | WRITE | 1111 0011 |
| 0FFF | 07 | WRITE | 1111 0011 |
| 0F06 | IN | A, (02) | 1111 0001 |
| 0202 | 4C | INPUT | 1111 0000 |
| 0F08 | LD | (HL), A | 1111 0000 |
| 2000 | 4C | WRITE | 1111 0000 |

На первый взгляд этот листинг покажется запутанным даже человеку, знающему ассемблерные коды микропроцессора Z80. Анализатор показывает на дисплее не только команды программы, но и те события на шинах адреса и данных, которые относятся к выполнению команд. Чтобы разобраться в листинге, пользователь должен знать особенности работы конкретного микропроцессора, в данном случае микропроцессора Z80.

Анализатор был запущен с использованием в качестве запускающего слова адреса 0D00, а информация с шины данных и с 8 внешних входов установлена на состояния «безразлично». В ячейках с адресами 0D00 и 0D01 хранились данные 3E и 06 соответственно. Управляющий микрокомпьютер был запрограммирован на дисассемблирование кодов микропроцессора Z80. На дисплее под заголовком КОП/ДАННЫЕ показана мнемоника LD A, 06, которая определяет загрузку в регистр A значения 06. Во второй строке дисассемблерной программы OUT (08) посылает это значение данных в выходной порт 08. В операции ВВ микропроцессора Z80 адрес порта ВВ выдается не только на младшие 8 линий шины адреса, но и дублируется на 8 старших линиях. Поэтому в следующей строке адрес порта (дублированный) показан как 0808 на шине адреса, а передаваемое в этот порт значение данных показано как 06 на линиях шины данных. Чтобы показать, что значение с шины данных записывается в выходной порт, после значения

данных (06 в этом случае) находится слово OUTPUT.

В четвертой строке листинга произведен вызов (CALL) подпрограммы из ячейки 0F06. Когда программа была написана в символической форме, а затем ассемблировалась в машинный код, адрес вызова, по-видимому, был символической меткой. Анализатор не может дисассемблировать такую информацию, так как она теряется в процессе ассемблирования; в лучшем случае анализатор может показать адрес первой команды в подпрограмме. Для хранения кода операции и 2-байтного адреса вызова в команде CALL требуется 3 байта памяти, поэтому следующая команда основной программы начинается по адресу 0D07. Этот адрес должен быть возвращен в программный счетчик по окончании подпрограммы, поэтому его необходимо сохранить в *стеке*. В данном примере предполагается, что при инициализации программы *указатель стека* был установлен на начальный адрес 1000 ЗУПВ. Адрес возврата (0D07) включается в стек путем декремента указателя стека (адрес 0FFF) с последующей записью в стек старшего байта (0D) программного счетчика. После этого вновь производится декремент указателя стека, и в стек записывается младший байт (07) программного счетчика. Передачи по шине, связанные с записью в стек, показаны в строках 5 и 6 листинга. В программный счетчик загружен начальный адрес подпрограммы 0F06, по которому, как видно в строке 7, находится команда ввода INPUT из порта BB с адресом 02. Считываемые из этого порта данные (16-ричное число 4C) и дублированный адрес порта (0202) видны в строке 8. Собственно данные загружаются в регистр *A* центрального процессора. Операция определяется на дисплее как считывание из входного порта словом INPUT, которое находится после значения данных 4C.

В последних строках листинга показано, что введенное в регистр *A* значение запоминается в ячейке памяти, адресуемой регистровой парой HL. Предполагается, что в регистровой паре HL находится адрес 2000, поэтому последняя строка показывает запись значения 4C в ячейку с адресом 2000.

Возможным недостатком приведенного формата можно считать наличие в нем передач по шине данных, которых нет в ассемблерном листнге отлаживаемой программы. Пользователь испытывает неудобство в процес-

се расшифровки содержимого дисплея, так как он не может просто идти по листингу, фиксируя возможные ошибки в программе. Ответом на этот аргумент служит то обстоятельство, что анализатор показывает важную информацию о реальных передачах на системных шинах, которая может сыграть ключевую роль в определении причины неправильной работы системы. Например, на дисплее видны используемые адреса стека, которые в силу ряда причин могут быть ошибочными. В предполагаемой области адресов стека микросхем памяти может не быть или указатель стека может адресовать другое устройство. Фактически используемые ячейки памяти могут хранить программу или данные. На дисплее видны реальные данные, передаваемые в порты ВВ, а они могут оказать решающую помощь в определении причины неправильной работы системы. В приведенном примере по операции считывания из входного порта 02 в ЦП передается значение 4С, а оно может быть не тем, которое ожидалось системой. Для проверки правильности работы порта 02 следует просмотреть его внешние входы. В порт могут подаваться неверные данные из-за какого-то внешнего отказа, но и сам порт может оказаться неисправным. Если к входному порту подключены тестовые переключатели, а к выходному порту — индикаторы, то анализатор позволяет проверить каналы ВВ информации.

Анализаторы логических состояний прошли эволюцию от встраиваемых в осциллографы модулей, появившихся в 1973 г., до автономных приборов, способных осуществлять дисассемблирование для конкретного микропроцессора (1977 г.). За четырехлетний период возможности анализаторов резко возросли, упростились их настройка и интерпретация результатов, а пользоваться приборами стало легко и удобно.

Анализаторы логических состояний в основном предназначены для упрощения отладки программ путем прослеживания и индикации смены состояний в алгоритмических автоматах, к которым относятся и микрокомпьютеры. Все операции в алгоритмическом автомате осуществляются синхронно с системной синхронизацией, которая применяется для регистрации и (вместе с другими управляющими сигналами, которые также синхронны с системной синхронизацией) для определения типа запоминаемой и индицируемой информации.

### 6.2. Анализаторы временных диаграмм

Иногда в системе возникают события, которые асинхронны относительно системной синхронизации, но которые могут вызвать ошибочное поведение системы. Такие события часто называются помехами и либо возникают из-за ошибок внутренней временной диаграммы системы, либо приходят от источников, находящихся вне вычислительной системы. Анализаторы логических состояний регистрируют информацию только синхронно с сигналами синхронизации проверяемой системы и не могут обнаружить появление помехи. Анализатор состояний опрашивает системные шины только по фронтам сигналов синхронизации. Изменение состояния любой линии шины, которое не приходится на фронты сигналов синхронизации, не регистрируется и не запоминается.

Для исследования временной диаграммы системы и регистрации помех разработаны анализаторы, в которых источником сигналов синхронизации и опроса служит их внутренний генератор сихронизации, работающий асинхронно с генератором синхронизации проверяемой системы. Подобный прибор называется *анализатором временных диаграмм;* он предназначен, главным образом, для показа временных диаграмм системы. Необходимость в таких анализаторах возникла по той причине, что осциллограф не может прослеживать и регистрировать сигналы со многих линий шины одновременно. Современные анализаторы временных диаграмм регистрируют и индицируют до 16 сигналов, а частота их внутренних генераторов синхронизации доходит до 200 МГц.

Следует подчеркнуть, что запомненная и в последующем индицируемая информация НЕ является действительным аналоговым представлением анализируемых сигналов. Информация запоминается во внутреннем ЗУПВ и индицируется в виде идеализированных прямоугольных сигналов. ЗУПВ обычно имеет 1024 или 2048 бит на каждый вход, а частота опроса варьируется от нижнего предела 2 Гц до принятого в анализаторе максимума. Из-за конечной емкости ЗУПВ приходится искать компромисс между задаваемой частотой опроса и объемом индицируемых событий в проверяемой системе. Если исследуется система с периодом синхронизации 500 нс и установлен период опроса 100 нс в анализаторе,

внутреннее ЗУПВ которого имеет 1024 ячейки, возможно зарегистрировать и индицировать 204 события, происходящих синхронно с сигналами синхронизации проверяемой системы. Если же период опроса уменьшен до 25 нс, можно зарегистрировать только 51 событие в проверяемой системе.

В логических системах существуют два типа асинхроиных временных измерений: *функциональный временной анализ и параметрический временной анализ.* Если мы заинтересованы только в том, что событие *А* происходит во время события *В*, имеет место фуикциональный временной анализ, который просто показывает корреляцию во времени различных событий. Когда же главная цель заключается в том, чтобы установить, что событие *А* длится определенный временной интервал или что событие *В* возиикает в течение определенного времени относительно события *А*, имеет место параметрический временной анализ, в котором важна точная временна́я диаграмма событий. Частота внутренней синхронизации в анализаторе временны́х диаграмм, предназначенном для параметрических временны́х измерений, должна быть выше частоты синхронизации в анализаторе, рассчитанном на функциональные временны́е измерения. Требование высокого быстродействия и применения соответствующих логических схем значительно удорожает параметрический временной анализатор по сравнению с функциональным временны́м анализатором.

Помехи или выбросы возиикают нз-за «гонок» в логических схемах, возникающих в результате плохой проработки отдельных схем системы, наводятся шумами в системе или от внешнего источнника. Импульс, продолжительность которого больше 5 нс, но меньше периода опроса, устаиовленного в приборе, квалифицируется анализатором временны́х диаграмм как помеха. Следовательно, при задании периода опроса 50 нс импульсы на любом из входов анализатора с длительностью в диапазоне 5—50 нс считаются помехами. В анализаторе временны́х диаграмм для помех и обычных системиых событий предусматриваются отдельные схемы регистрации и хранения. При индикации оба источника временной информации объединяются и помехи видны на временны́х диаграммах как яркие импульсы. В случае функционального временно́го анализа помехи индицируются в виде

ярких импульсов, ширина которых определяется установленным в приборе периодом опроса. Если, например, возникли две помехи с длительностью 30 и 70 нс, а период опроса составляет 100 нс, на индикаторе обе помехи имеют одинаковую длительность.

На практике анализаторы временны́х диаграмм применяются для исследования сигналов шины управления и операций ВВ, т. е. тех частей системы, в которых чаще всего возникают проблемы временно́го анализа. Приборы многих фирм, объявленные как совместимые с микропроцессором, рассчитаны на непосредственный интерфейс с относительно медленными микропроцессорными системами. Если такой прибор подключается к системе с более высоким быстродействием, необходимо обратить внимание на соответствие их временны́х характеристик. Чтобы прибор работал в пределах своих спецификаций, может потребоваться расширение управляющих импульсов и получение необходимых для прибора сигналов синхронизации из системной синхронизации с помощью делителя. Задержка распространения делителя и время установления прибора могут привести к «перекосу» синхронизации между прибором и системой. На этапе проектирования система кажется правильной, но неучтенный «перекос» синхронизации может вызвать неустойчивую работу. Для исследования помех, вызываемых гонками, и проверки реальных времен установления сигналов следует воспользоваться анализатором временны́х диаграмм.

Рассмотрим обнаружение помехи в схеме синхронного счетчика (см. рис. 4.2), в котором действует сигнал асинхронного сброса. Предположим, что счетчик формирует сигналы для других схем и возможность возникновения в нем помехи вызывает неправильную работу этих схем. Тогда анализатор временны́х диаграмм следует подключить к линии синхронизации и к трем выходам $Q_A$, $Q_B$ и $Q_C$. Пусть сигнал синхронизации, вызывающий изменение состояния счетчика, имеет частоту 2 Гц, а период — соответственно 500 мс. Чтобы зарегистрировать и индицировать все состояния счетчика, необходимо просмотреть полный цикл его работы длительностью 2,5 с. Предположим, что емкость ЗУПВ анализатора составляет 256 бит, так что задание периода опроса 100 мс позволит зарегистрировать в ЗУПВ как раз один полный цикл работы счетчика. Как и в анализаторах

логических состояний, в анализаторах временных диаграмм можно обусловить начало регистрации информации удовлетворением заданного критерия запуска. Настроим анализатор на начало регистрации данных, когда на всех трех выходах счетчика действуют состояния логического «0».

На обычном осциллографе заметить помеху сброса на выходе $Q_A$ практически невозможно, так как для просмотра всех состояний счетчика придется установить медленную развертку. Продолжительность помехи составляет всего 68 нс, поэтому ее нельзя заметить при периоде развертки 2,5 с. Однако анализатор временны́х диаграмм зафиксирует импульс сброса на выходе $Q_A$, так как его длительность больше 5 нс и меньше 100 мс, и покажет его как яркий сигнал помехи.

В большинстве анализаторов временны́х диаграмм допускается задание минимальной длительности помехи, что позволяет обнаруживать сигналы шириной больше 50 нс и меньше периода опроса. Регистрацию данных можно обусловить совокупностью событий, включая и появление помехи на одной или нескольких входных линиях. Такая возможность очень удобна в тех случаях, когда событие в системе возникает время от времени; исследователь может заниматься другими делами, а анализатор ожидает появления события. Анализаторы временны́х диаграмм можно настроить на регистрацию предзапусковых и задержанных событий, что вместе с возможностью запуска помехой превращает их в мощный контрольно-измерительный прибор. Возможно зарегистрировать, запомнить и индицировать для анализа события, которые привели к ошибке во временно́й днаграмме работы системы.

Удобной возможностью анализаторов временных диаграмм является подвижный курсор, с помощью которого при перемещении его по временны́м диаграммам вызывается индикация на экране времени задержки относительно запускающего набора. Пользуясь курсором, оператор может измерить временной интервал между событиями с точностью до периода внутренней синхронизации. Если, например, частота генератора синхронизации составляет 20 МГц, разрешающая способность измерений равна $\pm 50$ нс. Однако фактическая точность измерений временны́х интервалов зависит от числа десятичных знаков представления информации на экране. Если исполь-

зуются три десятичных знака, то при измерении интервала 15,648 мс будет показано 15,6 мс. Такой точности достаточно для большинства применений, за исключением параметрического анализа, при котором требуется точное значение.

### 6.3. Режимы индикации

В логическом анализаторе информацию можно индицировать несколькими способами, у каждого из которых имеются свои достоинства и недостатки. Режимы индикации совершенствовались в ходе эволюции самих приборов. Интересно отметить, что первые и более простые режимы индикации могут указать на отказ, который при представлении информацин в более компактной форме окажется завуалированным. Рассмотрим, например, такую последовательность:

| | | |
|---|---|---|
| 1101 | 0111 | D7 |
| 1011 | 0000 | B0 |
| 0100 | 0100 | 44 |

Слева информация представлена в двоичном формате, а справа — в 16-ричном. Она может относиться к 8 линиям шины данных, в которой из-за отказа на линии $D_3$ постоянно действует иизкий уровень. Отказ очевиден в двоичном формате, а в 16-ричных кодах не так заметен.

Для логических анализаторов разработано семь режимов индикации, хотя ни в одном приборе все вместе они не реализованы.

#### 6.3.1. Режим карты

Предположим, что спроектирована и построена микропроцессорная система, а для ее проверки написана короткая тест-программа. Первое, в чем нужно убедиться после включения системы, — это то, что машина обращается по адресам тест-программы, а не пытается из-за отказа выполнить несуществующую программу в другой области памяти. Для определения поведения системы достаточно просмотреть только состояния линий шины адреса, так как в данной ситуации нас интересуют только адреса, а не передачи по шине данных. В большинстве первых логических анализаторов и в некоторых

более поздних моделях предусмотрен режим карты, который отображает действия в шине адреса. В режиме карты логические слова длиной до 16 бит однозначно отображаются в виде отдельных точек на экране. Весь экран может представлять 65 536 возможных значений адреса 16-битной шины, причем 0000 изображается точкой в верхнем левом углу экрана, а FFFF — точкой в нижнем правом углу. Когда проверяемая система использует любой адрес, он появляется в соответствующей позиции как яркая точка. Если сформировать сигналы развертки между точками, то по карте на экране можно проследить процессы в шине адреса. В анализаторе 1607 S фирмы Hewlett-Packard линия развертки шире у первого адреса и сужается к следующему. На рис. 6.8 показано, каким образом на экране представляется все адресное пространство 8-битного микрокомпьютера.

Показанные действия представляют собой изображение, которое получается, когда программа, работающая в области младших адресов, вызывает подпрограмму из области с большими адресами. Подобные действия характерны, например, для микрокомпьютера, который периодически опрашивает клавиатуру и находится в цикле до нажатия клавиши. Для определения точного значения адреса переключатели, с помощью которых устанавливается запускающий адрес, используются для позиционирования курсора на экране. Положения переключателей соответствуют адресу той точки, с которой

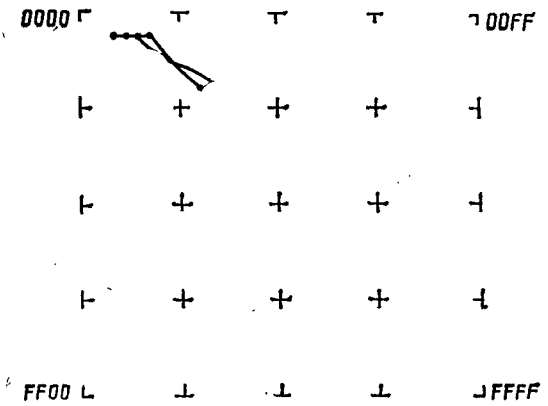



**Рис. 6.8.** Квадранты режима карты в 16-ричном представлении

совмещен курсор. Приведенное на рис. 6.8 изображение экрана не позволяет точно определить адрес, но каждый из 16 его квадрантов можно расширить на весь экран и точно определить адрес с помощью курсора. Квадрант представляет всего 4096 адресных точек, и каждую из них можно локализовать на экране курсором. Обычно с помощью органов управления масштабированием квадрант с курсором расширяется на весь экран; затем курсор перемещается на интересующую оператора точку, и положения переключателей определяют ее адрес. На рис. 6.9 показаны примеры событий на шине адреса, происходящих в области младших адресов, и позиционирование курсора на одну из точек с целью определения точного значения адреса.

На рис. 6.9 значение адреса точки, с которой совмещен курсор, равно 028D. Расширенный квадрант является первым квадрантом в адресном пространстве памяти системы и ограничен 16-ричными адресами 000...0FFF.

Режим карты весьма удобен в процессе разработки системы, так как дает общую картину поведения системы и показывает любые аномалии. Однако во многих более поздних и более сложных логических анализаторах такой режим не предусмотрен.

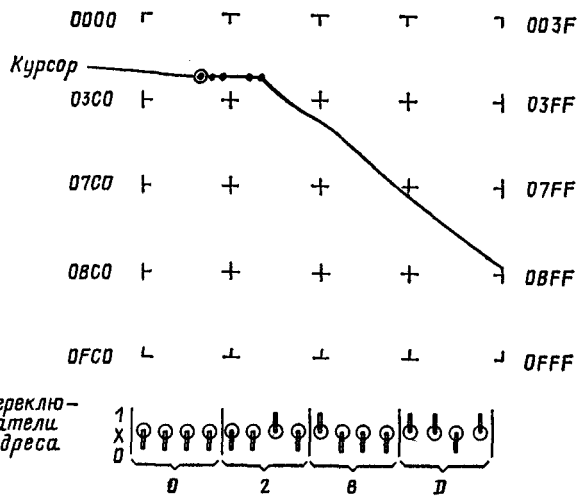



**Рис. 6.9.** Режим расширенной карты с показом одного из 16 квадраитов

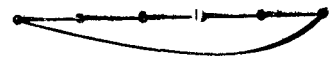


Рис. 6.10. Изображение карты счетчика с 6 состояниями

В режиме карты позиция точки соответствует значению ее двоичного адреса в карте адресов, а яркость определяет относительную частоту появления. Линия развертки между двумя точками означает изменение двоичного набора, причем линия толще в начальной точке и сужается к конечной точке. По существу, режим карты является графическим способом представления двоичных событий в системе, и он не ограничивается только показом адресов. С помощью режима карты можно посмотреть, например, работу двоичного счетчика, и полученное изображение образует уникальный «узор» для счетчика.

На рис. 6.10 показано изображение, полученное с выходов двоичного счетчика на 6 состояний, который циклически проходит все 6 состояний, а затем возвращается в нулевое состояние. Изображение уникально для каждого счетчика, и его можно рассматривать как «отпечатки пальцев» счетчика. Оператор может приобрести опыт по распознаванию характеристических «узоров» микросхем в режиме карты и идентифицировать по ним либо конкретную схему, либо поведение системы. Режим карты применяется довольно редко, так как он показывает поведение шины адреса необычным образом, и для осмысленного восприятия этой информации необходим опыт. Тем не менее трудно придумать что-то лучшее для быстрой идентификации событий в шине адреса, чем режим карты. Он имеется в логических анализаторах 1607S фирмы Hewlett-Packard, 7D01 фирмы Tektronix и 1650-D фирмы Gould Biomation.

#### 6.3.2. Двоичный и группированный двоичный режим

Как только в логических анализаторах появилось внутреннее ЗУПВ для хранения данных, воспринятых от проверяемой системы, стало естественным представлять эти данные на индикаторе в двоичном формате. Каждый адрес и соответствующее значение данных индицировались на экране в виде логических 0 и 1, причем запускающие слова и предзапусковые значения находились вверху экрана, а задержанные значения — ниже их. Для

удобства восприятия информации двоичные цифры группировались по три или четыре и считывались как 8- или 16-ричные значения. Такой режим характерен для анализаторов логических состояний и показывает смену состояний в системе; типичное изображение информации на экране выглядит следующим образом.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0101 | 0111 | 1101 | 0100 | 0101 | 0111 | 1101 | 0100 |
| 0010 | 1011 | 1010 | 1000 | 0110 | 0011 | 1000 | 1100 |
| 0001 | 0100 | 0101 | 0000 | 0011 | 0000 | 0001 | 1000 |
| 1100 | 0100 | 0100 | 0110 | 0100 | 1100 | 0110 | 0100 |
| 0110 | 0011 | 1000 | 1100 | 1010 | 0111 | 1100 | 1010 |
| 0011 | 0000 | 0001 | 1000 | 1101 | 0011 | 1001 | 0110 |
| 1001 | 1000 | 0011 | 0010 | 1110 | 1000 | 0010 | 1110 |
| 0100 | 1100 | 0110 | 0100 | 0111 | 0100 | 0101 | 1100 |
| 1010 | 0111 | 1100 | 1010 | 1011 | 1011 | 1011 | 1010 |
| 1101 | 0011 | 1001 | 0110 | 1101 | 1100 | 0111 | 0110 |
| 1110 | 1000 | 0010 | 1110 | 0110 | 1111 | 1110 | 1100 |
| 0111 | 0100 | 0101 | 1100 | 0011 | 0111 | 1101 | 1000 |
| 1011 | 1011 | 1011 | 1010 | 0001 | 1011 | 1011 | 0000 |
| 1101 | 1100 | 0111 | 0110 | 0000 | 1100 | 0110 | 0000 |
| 0110 | 1111 | 1110 | 1100 | 1000 | 0111 | 1100 | 0010 |
| 0011 | 0111 | 1101 | 1000 | 1100 | 0011 | 1000 | 0110 |

Левый блок двоичных цифр показывает зарегистрированную в некоторой системе смену состояний; цифры сгруппированы по 4, чтобы оператор мог преобразовать их в 16-ричные значения. Для удобства восприятия изображение разделено и по вертикали на группы из 4 строк. Верхняя строка дает 16-ричное значение 57D4 и может представлять собой адрес, зарегистрированный в микропроцессорной системе.

В целом каждая строка на экране занимает левый и правый блоки и представляет 32-битное слово, которое состоит из 16-битного адреса, кода с 8-битной шины данных и 8 внешних линий. В приведенном примере левый блок содержит информацию с шины адреса, а правый — коды с шины данных и состояния внешних линий.

Однако в большинстве анализаторов, имеющих такой режим индикации, экран разделяется на две отдельные части. В правой части находятся данные, воспринятые от проверяемой системы в некоторый период времени, а в

левой — данные, зарегистрированные в тех же самых линиях позднее, но с одинаковыми условиями запуска. Когда имеются любые различия, цифры в левой части светятся ярче, показывая изменение состояния по сравнению с даиными, которые зарегистрированы в правой части. Выше представлены именно такие события: в правой части показаны воспринятые от системы данные, а в левой — данные от тех же источников, но зарегистрированные позднее. Выделенные жирным шрифтом цифры показывают те биты, которые в обеих частях различаются. Такой режим индикации наиболее удобен в тех ситуациях, когда проверяются входные значения от цифровых датчиков, например, от концевых переключателей или детекторов сближения. Здесь оператор заинтересован, главным образом, в двоичных состояниях сигналов, поэтому двоичный формат (сгруппированный или нет) идеально подходит для этой цели. Изменения состояний входов легко видеть по более ярким цифрам на экране, которые обозначают срабатывание датчиков.

Как упоминалось выше, двоичные индикаторы удобны для контроля замыкания линий на землю (постоянный 0) или на питание (постоянная 1). В этом смысле они действуют как логический пробник, который хранит много прошлых событий, а не просто показывает текущее событие.

Сравнение двух наборов данных и выделение яркостью отличающихся бит реализованы в логических анализаторах LA5000 фирмы Gould Biomation, 1607S фирмы Hewlett-Packard и 7D01 (с форматтером дисплея DF1) фирмы Tektronix.

#### 6.3.3. Шестнадцатеричный формат

Следующим эволюционным шагом индикации смены состояний в анализаторе логических состояний было преобразование сгруппированных двоичных цифр в восьмеричные или шестнадцатеричные символы. Восьмеричные цифры в основном применялись в миникомпьютерах, а в большинстве микропроцессорных систем принято 16-ричное кодирование. Индикация в 16-ричном формате используется при наличии листинга машинного кода проверяемой программы, что позволяет получить однозначное соответствие между изображением на экране и листингом.

Предположим, например, что написана подпрограм-

ма задержки, начинающаяся по адресу 0F00, и ее ассемблерный листинг имеет следующий вид:

```
МЕТКА     КОП/ОПЕРАНД          КОММЕНТАРИЙ
DELAY:    LD      B, FF        ; загрузить 255 в регистр
                               ; B
DEL:      LD      C, FF        ; загрузить 255 в регистр
                               ; C
LOOP:     DEC     C            ; декремент регистра C
          JP      NZ, LOOP     ; если регистр C≠0, пе-
                               ; рейти к LOOP
          DEC     B            ; декремент регистра B
          JP      NZ, DEL      ; если регистр B≠0, пе-
                               ; рейти к DEL
          RET                  ; если регистр B=0, воз-
                               ; врат
```

Такая подпрограмма задержки, написанная в мнемониках микропроцессора Z80, является типичным примером программы формирования небольшой временной задержки. После ассемблирования в машинный код, начиная с адреса 0F00, программа приобретает следующую форму:

```
АДРЕС     КОП/ОПЕРАНД
0F00      06 FF
0F02      0E FF
0F04      0D
0F05      C2 04 0F
0F08      05
0F09      C2 02 0F
0F0C      C9
```

Когда эта подпрограмма вызывается из основной программы, внутренний цикл с меткой LOOP выполняется 255 раз до достижения нуля в регистре *C*. После этого выполняется внешний цикл с меткой DEL, в котором производится декремент регистра *B* и повторение внутреннего цикла еще 255 раз. В результате получается 255 вызовов внутреннего цикла, который сам выполняется 255 раз. Всего внутренний цикл вызывается несколько более 65 000 раз, и при частоте синхронизации 4 МГц получается задержка примерно в 230 мс.

Если запустить логический анализатор по адресу 0F00, то при вызове подпрограммы на экране появится следующее изображение:

| АДРЕС | КОП/ОПЕРАНД |
|---|---|
| 0F00 | 06 |
| 0F01 | FF |
| 0F02 | 0E |
| 0F03 | FF |
| 0F04 | 0D |
| 0F05 | C2 |
| 0F06 | 04 |
| 0F07 | 0F |
| 0F04 | 0D |
| 0F05 | C2 |
| 0F06 | 04 |
| 0F07 | 0F |
| 0F04 | 0D и т. д. |

На экране видны передачи по системным шинам кодов операций и операндов каждой команды. Когда ЦП выбирает всю команду условного перехода из ячейки 0F07, декремент регистра *C* еще не дает нуля, поэтому программа возвращается к ячейке 0F04 для повторного выполнения команды декремента регистра *C*. Ограниченная емкость ЗУПВ анализатора означает, что невозможно запомнить и показать всю подпрограмму, если не сделать начальные значения в регистрах *B* и *C* небольшими. Можно повторно запустить анализатор по адресу 0F08, чтобы проверить окончание внутреннего цикла при достижении нуля в регистре *C*. Можно также запустить анализатор по команде возврата в ячейке 0F0C и указать некоторое предзапусковое значение, чтобы просмотреть завершение подпрограммы.

Индицируемая информация соответствует листингу машинного кода, и ее можно использовать для проверки правильности ввода и выполнения программы. Однако без указания дополнительных признаков данный способ проверки программы оказывается весьма громоздким и продолжительным. Другие возможности анализаторов, связанные с обусловливаиием запускающих слов, которые упрощают проверку циклической программы с большим числом повторений, рассмотрены в § 6.4.

#### 6.3.4. Дисассемблерный формат

В иастоящее время наиболее совершенным форматом индикации смены состояний в конечном автомате является представление информации в дисассемблерной фор-

ме. Так как в каждом микропроцессоре мнемоники команд различны, в анализаторе требуется использование платы персонификации для того микропроцессора, который работает в проверяемой системе. Эквивалентный дисассемблерный формат для примера из п. 6.3.3 имеет следующий вид:

| АДРЕС | КОП/ОПЕРАНД | | ВНЕШНИЕ |
|---|---|---|---|
| 0F00 | LD | B, FF | 0000 0000 |
| 0F02 | LD | C, FF | 0000 0000 |
| 0F04 | DEC | C | 0000 0000 |
| 0F05 | JP | NZ, 0F04 | 0000 0000 |
| 0F04 | DEC | C | 0000 0000 |
| 0F05 | JP | NZ, 0F04 | 0000 0000 |
| 0F04 | DEC | C | 0000 0000 |

и т. д.

Выполнение программы теперь представлено в более удобной форме — вместо машинных кодов команд индицируются их мнемоники. Очевидно, анализатор не может интерпретировать метки, имеющиеся в исходной программе, поэтому в командах условных переходов и вызовов подставлены фактические значения адресов.

Хотя этот формат облегчает интерпретацию информации на экране, в нем сохраняются проблемы введения дополнительных признаков для контроля общих временных соотношений и числа повторений циклов.

Анализатор 1611A фирмы Hewlett-Packard допускает персонификацию под несколько микропроцессоров и может отображать их дисассемблерные коды.

### 6.3.5. Индикация временны́х диаграмм

Некоторые анализаторы, например модели 9100-D и 920-D фирмы Gould Biomation, могут индицировать только временны́е диаграммы для параметрического и функционального временно́го анализа. Большинство же анализаторов, рассчитанных на индикацию временны́х диаграмм, имеет режимы индикации в группированном двоичном и 16-ричном форматах, а также режим карты. Например, анализатор 1650-D фирмы Gould Biomation имеет 16 каналов и может индицировать информацию в группированном двоичном и 16-ричном форматах или в режиме карты. Анализатор 1615A фирмы Hewlett-Packard

показывает смену состояний в двоичном, восьмеричном, десятичном или 16-ричном форматах и может использоваться как 8-канальный прибор для функционального временно́го анализа.

Индикация временны́х диаграмм играет важную роль в решении проблем распределения во времени сигналов, которые связаны с линиями шины управления и (или) внешними входами. Например, анализатор может зарегистрировать и показать помехи, вызывающие случайные прерывания, даже если они возникают только время от времени.

#### 6.3.6. Форматы специального кодирования

При анализе асинхронной последовательной линии передачи данных, связывающей компьютер с периферийным устройством, например принтером или видеотерминалом, приходится учитывать, что данные передаются символами, а биты символов передаются последовательно. Для передачи по линии не просто числовых данных применяется способ кодирования, при котором двоичный набор символа определяет его как букву, цифру или управляющее слово. Наибольшее распространение получило кодирование в *Американском стандартном коде для обмена информацией* (ASCII). Большинство ранее рассмотренных анализаторов рассчитаны на восприятие данных в параллельном формате, а здесь в кодах ASCII информация передается последовательно. Были разработаны специальные анализаторы, которые декодируют последовательные двоичные наборы и преобразуют их в эквивалентные слова кода ASCII для последующей индикации. Самым распространенным электрическим и механическим стандартом последовательного интерфейса является стандарт EIA RS232C. Выпускаются анализаторы, например, модель 1640B фирмы Hewlett-Packard, предназначенные для измерений и взаимодействия с последовательными линиями, так что линии можно полностью испытать и проверить.

Аналогичные проблемы возникают при проверке параллельного интерфейса IEEE-488, в котором взаимодействие между системами осуществляется по 16 линиям. Адреса и данные передаются по 8 линиям, а остальные линии применяются для управления. Все символы обычно передаются по 8-битной шине в коде ASCII, а для

управления зарезервированы 3 линии квитирования. Остальные линии идентифицируют информацию на шине как адрес или данные и обеспечивают общее управление системой. Передачи по шине описываются специальными мнемоническими кодами, которые можно проанализировать и показать в мнемоническом формате, пользуясь такими анализаторами, как 1602A фирмы Hewlett-Packard, или 7D01 (со специальным форматтером дисплея DF2) фирмы Tektronix.

### 6.4. Дополнительные возможности логических анализаторов

Первые логические анализаторы были системами с «жесткой логикой», которые настраивались вручную с пульта управления путем задания таких параметров, как запускающий адрес и код на шине данных. Возможности изображения на экране ограничивались группированным двоичным форматом или временными диаграммами и слабо учитывали требования эргономики. С появлением в анализаторе микропроцессора в качестве управляющего элемента стало возможным реализовать режим «подсказки» пользователю при вводе настроечных данных с помощью сообщений, запрашивающих спецификации прослеживания, например состояния шины адреса и др. Во многих современных логических анализаторах применяется принцип «меню», в котором на экран выводится несколько изображений, запрашивающих у пользователя ввод необходимой для анализатора информации, прежде чем он начинает регистрацию данных.

| ЗАПУСК | АДРЕС | ДАННЫЕ | ВНЕШНИЕ |
|---|---|---|---|
| | 0F00 | XX | XXXX XXXX |

СТРОКА 0

| АДРЕС | КОП/ДАННЫЕ | | ВНЕШНИЕ |
|---|---|---|---|
| 0F00 | LD | B, FF | 0000 0000 |
| 0F02 | LD | C, FF | 0000 0000 |
| 0F04 | DEC | C | 0000 0000 |
| 0F05 | JP | NZ, 0F04 | 0000 0000 |
| 0F04 | DEC | C | 0000 0000 |
| 0F05 | JP | NZ, 0F04 | 0000 0000 |
| 0F04 | DEC | C | 0000 0000 |

Показанное изображение характерно для анализатора логических состояний 1611A фирмы Hewlett-Packard, оснащенного модулем персонификации для дисассемблирования мнемоник микропроцессора Z80. Анализатор запущен с адреса 0F00, причем для кода в шине данных по адресу 0F00 введено состояние «безразлично», а 8 внешних линий также не определяют условие запуска. Анализатор 1611A имеет несколько клавишных блоков, с которых вводятся настроечные параметры и которые дают возможность пользователю начать регистрацию смены состояний в системе. Рамка, в которую заключена первая строка, показывает, что она обычно дается в инверсном (по светности) изображении, чтобы выделить запускающее слово.

Если в систему были введены условия предзапуска, номер в верхней строке (строка 0) покажет число запрошенных событий. На экране будут показаны эти события с сохранением запускающего слова в инверсном изображении. Запускающее слово появится на экране ииже, после заданного числа предзапусковых событий. Приведенный номер строки показывает, что верхняя строка на экране является запускающим словом, а так как анализатор 1611A одновременно выводит только 16 строк, то клавиша «рулона» позволяет просмотреть все 64 зарегистрированных события. По мере движения строк номер строки увеличивается, показывая, что верхняя строка на экране находится в памяти после запускающего слова.

#### 6.4.1. Подсчет событий

При отладке программ иногда важно определить, сколько раз появилось конкретное событие. Например, может потребоваться знать, сколько раз фактически выполнялась команда декремента регистра *C* (DEC C) при вызове подпрограммы задержки. Многие анализаторы дают возможность подсчитать, сколько раз возникло запускающее событие между задаваемыми словами «разрешен» и «запрещен». Если, например, анализатор разрешается по адресу 0F00 и запрещается по адресу 0F0C и запускающим событием указан адрес 0F04, анализатор подсчитает и покажет, сколько раз адрес 0F04 появился между первым и последним адресами подпрограммы задержки. Для производства подобных измерений клавиатура анализатора имеет несколько наборов

запускающих клавиш, обозначенных РАЗРЕШЕНИЕ ЗАПУСКА, ЗАПУСК и ЗАПРЕЩЕНИЕ ЗАПУСКА. Каждый набор клавиш можно использовать для задания состояний шины адреса, шины данных и внешних линий в качестве признаков, причем состояния «безразлично» в этих параметрах означают, что они не участвуют в определении запуска. Обычно используется только информация шины адреса.

| | АДРЕС | ДАННЫЕ | ВНЕШНИЕ |
|---|---|---|---|
| РАЗРЕШЕН | 0F00 | | |
| ЗАПУСК | 0F04 | | |
| ЗАПРЕЩЕН | 0F0C | | |

----------ОСТАНОВЛЕН ----

| АДРЕС | КОП/ДАННЫЕ | ВНЕШНИЕ |
|---|---|---|
| СЧЕТ | 65 025 | ЗАПУСКОВ |
| МАКС | 65 025 | ЗАПУСКОВ |
| МИН | 65 025 | ЗАПУСКОВ |

Выше пунктиром линии показаны настроечные параметры, а сама линия означает, что анализатор зарегистрировал требуемую информацию и остановился. Анализ был разрешен, начиная с адреса 0F00, причем для значений шины данных и внешних линий были введены состояния «безразлично». Обычно они не индицируются, поэтому, если для шины данных и внешних линий информация не вводится, анализатор по умолчанию принимает состояния «безразлично». Разрешающий адрес соответствует первому адресу подпрограммы, а с помощью клавиши запрещения анализа был введен конечный адрес 0F0C. Теперь анализатор может подсчитать, сколько раз запускающее слово появляется между граничными адресами 0F00 и 0F0C. Комаида DEC C находится по адресу 0F04, который введен в анализатор как запускающее слово с помощью соответствующих клавиш. В анализаторе 1611A для инициирования подсчета событий нужно нажать специальную клавишу СЧЕТ ЗАПУСКОВ; счет начнется, когда впервые встретится запускающий адрес, и закончится при обнаружении запрещающего адреса.

На экране показано, что в подпрограмме задержки комаида DEC C выполнялась 65 025 раз. Еще две строки

показывают минимальное и максимальное числа зарегистрированных запускающих событий. Пользуясь клавишей СЧЕТ ЗАПУСКОВ, можно подсчитать число запускающих событий для нескольких вызовов подпрограммы, и имеющиеся различия будут показаны в строках МАКС и МИН. Значения МАКС и МИН представляют интерес, когда между разрешающим и запрещающим запускающими словами имеется команда условного перехода, которая может обойти запускающее событие, в результате чего могут получиться различные значения подсчитанных событий.

### 6.4.2. Измерения временны́х интервалов в анализаторе логических состояний

При анализе работы системы иногда требуется знать с приемлемой точностью времена выполнения определенных фрагментов программы. Примером может служить время выполнения подпрограммы задержки, которое из-за небольшого отличия частоты системной синхронизации от номинального значения может отличаться от требуемого. Подпрограмму задержки можно использовать для формирования разрешающего строба в аналого-цифровом преобразователе; например, в преобразователе AD574 время преобразования аналогового входа составляет 25 мкс. В преобразователь выдается команда преобразования, а затем вызывается подпрограмма задержки на 25 мкс, определяющая время преобразования до считывания из преобразователя цифрового кода. Если из-за ухода частоты системной синхронизации задержка составит только 20 мкс, считывание из преобразователя даст бессмысленный результат.

Другим примером измерения временно́го интервала является оценка времени между двумя внешними событиями, которое из-за физической природы процесса должно находиться в заданных пределах. События могут соответствовать срабатыванию исполнительного органа и восприятию его реакции от детектора сближения. Процессор необходимо перевести в цикл ожидания срабатывания детектора сближения, после чего система переходит к следующим действиям. Параллельно с циклом ожидания работает «таймер-сторож» для контроля нахождения реакции в заданном диапазоне. В качестве таймера в микропроцессорных системах применяется

микросхема программируемого таймера, которая программируется на фиксированную временну́ю задержку и прерывает работу ЦП по истечении заданного времени. Если детектор сближения срабатывает до истечения интервала времени на таймере, то его дальнейшая работа запрещается и продолжается выполнение программы. Во многих системах с механическими исполнительными органами максимально допустимое на срабатывание время неизвестно, поэтому в таймер невозможно загрузить точное значение временно́го интервала. Произведя измерения реального временно́го интервала в работающей системе, можно найти максимально допустимый временной интервал и использовать его в качестве константы для таймера. Если срабатывание длится дольше заданного максимума, таймер прерывает работу ЦП и сигнализирует о наличии в системе механического отказа.

Измерения временны́х интервалов реализуются в анализаторе логических состояний посредством задания разрешающего и запрещающего запускающих слов, а фактическое запускающее событие определяется состояниями «безразлично». Для инициирования измерения применяется клавиша ВРЕМЕННОЙ ИНТЕРВАЛ; анализатор начинает работать, когда входы от системы совпадают с параметрами разрешающих клавиш, и останавливается, когда удовлетворяются условия запрещения запуска. Точность измерения обычно определяется частотой внутреннего генератора синхронизации, и типичное ее значение равно 1 мкс.

| | АДРЕС | ДАННЫЕ | ВНЕШНИЕ |
|---|---|---|---|
| РАЗРЕШЕНИЕ | 0F00 | | |
| ЗАПУСК | XXXX | | |
| ЗАПРЕЩЕНИЕ | 0F0C | | |

..... ОСТАНОВЛЕН .....

| АДРЕС | КОП/ДАННЫЕ | | ВНЕШНИЕ |
|---|---|---|---|
| ВРЕМЯ | | 228 928 | МИКРОСЕКУНД |
| МАКС | | 228 928 | МИКРОСЕКУНД |
| МИН | | 228 928 | МИКРОСЕКУНД |

В приведенном примере измерения временно́го интервала для подпрограммы задержки период синхронизации

предполагается равным 250 нс, он характерен для микропроцессора Z80, работающего с частотой синхронизации 4 МГц. Измеренное время выполнения подпрограммы равно 0,229 с; оно согласуется с результатом подсчета событий. Как видно, на экран выводятся максимальное и минимальное значения временно́го интервала.

В примере с подпрограммой задержки маловероятно получить существенные различия в ее времени выполнения, но в примере ожидания системой сигнала с внешнего входа различия во временны́х интервалах могут быть существенными.

```
ACTUATE:    LD      A, 08       ; установить бит испол-
            OUT     (04), A     ; нительного органа в
                                ; выходном порту 4
INPUT:      IN      A, (02)     ; считать входной порт
            AND     01          ; для проверки детекто-
                                ; ра сближения, подклю-
                                ; ченного к биту 0 пор-
                                ; та 2
            JP      NZ, INPUT   ; повторять цикл до ус-
                                ; тановки в «1» бита 0
```

Ассемблерная программа микропроцессора Z80 включает исполнительный орган, подключенный к выходному порту 4, а затем ожидает в цикле считывания входа, когда сработает детектор сближения или граничный переключатель, подключенный к младшему биту входного порта 2. На практике до этого фрагмента осуществляется программирование таймера на время операции, и он прерывает ЦП, если отведенное время истекло. Когда сигнал на входе появляется до истечения заданного времени, последующие команды запрещают прерывание ЦП от таймера.

Анализатор следует запустить по коду адреса, относящемуся к метке ACTUATE, и запретить по адресу команды, находящейся после команды условного перехода. Потребуется произвести несколько измерений, чтобы оценить время, на которое следует запрограммировать таймер для уверенного обнаружения отказа в системе.

#### 6.4.3. Диапазоны запуска и условия запуска

Во многих анализаторах вместо определения единственного запускающего набора разрешается задать запускающее событие как значение адреса, находящееся

в диапазоне адресов. Может потребоваться зарегистрировать любые события, возникающие между адресом 0100 и адресом 0200; эти адреса можно ввести в анализатор с клавиатуры, определив запускающее окно. Первый адрес, зарегистрированный анализатором и находящийся между заданными значениями, действует в приборе как запускающее слово.

В анализаторе часто предусматривается возможность регистрации и запоминания информации от проверяемой системы только после того, как запускающее слово появится заданное число раз. Такой режим применяется в тех случаях, когда число проходов по циклу известно, а требуется показать только последний проход и последующие команды программы. Предположим, например, что цикл выполняется 20 раз, а затем выполняется следующий фрагмент программы. Необходимо ввести число появлений запускающего слова, равное 19, а затем запустить анализатор. Информация будет регистрироваться от проверяемой системы лишь после 19 появлений запускающего слова, поэтому на экране будут показаны только последний проход по циклу и последующие команды программы.

#### 6.4.4. Предзапуск и постзапуск

Обычно анализатор логических состояний регистрирует и индицирует входной набор, соответствующий заданному запускающему слову, и последующие события вплоть до заполнения его внутренней памяти. Многие анализаторы могут запоминать всего 64 события, поэтому у пользователя оказывается мало информации о работе системы. Для расширения возможностей анализатора разрешается определять несколько предзапусковых событий вплоть до предела, определяемого емкостью его внутренней памяти. Если, например, анализатор может запомнить 64 события, то можно сохранить и показать до 63 предзапусковых событий, происходящих до запускающего слова. Такая возможность удобна при поиске неисправности, когда возникло случайное ошибочное событие, и желательно проследить те события, которые привели к ошибке.

Постзапуск вводит в измерения задержку, давая пользователю возможность определять число событий в системе, которые могут произойти после восприятия ана-

лизатором запускающего набора, прежде чем прибор начнет регистрировать информацию. В примере с подпрограммой задержки можно определить значение задержки 65 024 и задать запускающее слово по адресу 0F04. В этом случае подпрограмма задержки выполнит все циклы, кроме последнего, прежде чем анализатор начнет регистрировать информацию. На первый взгляд данная возможность аналогична определению числа появлений запуска, которое должно произойти до начала регистрации данных; однако значение задержки обычно гораздо больше того числа появлений, которое можно определить, и обеспечивает большую область действия эффективной цифровой задержки.

#### 6.4.5. Модули персонификации

В некоторых анализаторах логических состояний предусмотрены модули персонификации для конкретных микропроцессоров; применение этих модулей позволяет отображать на экране дисассемблерный код в мнемониках микропроцессора и наблюдать больше сигналов шины управления. Дополнительные условия запуска в модуле персонификации дают возможность определять те передачи по шине, которые необходимо запоминать и индицировать. Например, можно зарегистрировать и показать на экране только передачи ВВ. Можно обусловить регистрацию только операций записи, а если при этом задать операции ВВ, то анализатор покажет только операции записи в выходные порты.

Основные сигналы шины управления обычно индицируются на светодиодах, встроенных в модуле, поэтому пользователь может сразу увидеть, что правильному функционированию системы препятствует отказ в управляющей линии. Обычно индицируется отсутствие синхронизации, а также состояния линий RESET и WAIT, которые могут не позволить микропроцессору выполнять любые программы.

В настоящее время анализатор логических состояний с модулем персонификации для проверяемой микропроцессорной системы оказывается наиболее простым и быстрым средством отладки программного обеспечения системы. Кроме того, изучив работу с прибором и освоив формат индикации, с помощью такого анализатора можно решить и проблемы ввода-вывода информации.

### 6.4.6. Возможности анализаторов временны́х диаграмм

Анализаторы временны́х диаграмм обычно показывают 4, 8 или 16 диаграмм одновременно. Как и анализаторы логических состояний, они обеспечивают гибкий набор условий запуска до начала регистрации данных от системы. Обычно в качестве такого условия используется состояние любой входной линии. Кроме того, для расширения спецификации запуска можно использовать помехи на любом числе входных линий.

Компромисс между «глубиной запоминания» и частотой синхронизации означает, что анализатор временны́х диаграмм должен обеспечивать широкий диапазон периода синхронизации — от 20 нс до 0,5 с.

Анализатор временны́х диаграмм должен обладать возможностью точных временны́х измерений на экране с помощью подвижного курсора. Во многих анализаторах имеется возможность масштабирования индикации для повышения точности измерений.

### 6.4.7. Принадлежности

Для контроля происходящих на шине событий к проверяемой системе необходимо подключать проводники. Анализатор широкого назначения имеет несколько переходных устройств для сбора данных от проверяемой системы, причем к каждому устройству обычно подключаются 9 проводников. Для сбора информации с шин системы и внешних линий применяются до 4 переходных устройств. Каждый проводник заканчивается небольшим зажимом, который можно подключить к выводу микросхемы, не закорачивая соседних выводов. При необходимости контроля 32 линий подключение зажимов становится утомительной и трудной задачей.

Для анализатора широкого назначения эффективное решение заключается в том, чтобы «надеть» на микропроцессор клипсу и вывести все его выводы на доступную сверху площадку. Для конкретной системы проводники от анализатора можно припаять непосредственно к клипсе.

В анализаторах с модулями персонификации микропроцессор обычно удаляется из проверяемой системы и заменяется насадкой с корпусом DIP, соединенной плоским кабелем с переходным устройством, которое содер-

жи́т *разъем с нулевым усилием стыковки* (ZIF), куда и устанавливается микропроцессор. За исключением внешних линий, все необходимые для анализатора входы можно взять от микропроцессора, что устраняет необходимость применения гибких проводников, соединяющих анализатор с проверяемой системой. Одно из главных преимуществ размещения микропроцессора проверяемой системы в переходном устройстве заключается в том, что анализатор может не только контролировать проверяемую систему, но и управлять ею. Переводя микропроцессор в состояние HOLD от анализатора, можно осуществнть выполнение программы по отдельным командам, что иногда оказывается невозможным при работе только с проверяемой системой.

## 7. Сигнатурный анализ

Аналоговые схемы проверяются путем подачи известных тест-наборов и контроля каждого узла в тракте распространения сигнала с помощью осциллографа. При обнаружении нестандартного сигнала подозрение падает на ту часть схемы, которая управляет узлом, и именно она исследуется более тщательно. В принципе таким же образом можно проверять и цифровые схемы: подавать известные тест-наборы и контролировать каждый узел в тракте распространения сигнала. Однако цифровые системы радикально отличаются от аналоговых систем не только самой природой сигналов, но и наличием гораздо большего числа сигнальных входов. Если бы цифровой системой можно было управлять таким образом, что на всех ее входах действовал четко определенный тест-набор, то каждый узел можно было бы проверить на фиксированный набор событий. В любом узле при каждом выполнении стимулирующей программы возникает один и тот же набор, и его можно использовать для проверки правильной работы узла. Если измеряемый набор отличается от ожидаемого, следует подозревать наличие отказа между данным узлом и стимулируемыми точками системы. При заданном фиксированном тест-наборе измеряемый набор в любом узле оказывается уникальным, и его можно использовать для контроля по принципу «проходит/не проходит». Измеряемая реакция узла на известный тест-набор назы-

вается «сигнатурой» (т. е. подписью) по аналогии с уникальностью рукописных подписей у людей. Рассмотренный принцип лежит в основе целой области цифрового тестирования, называемой *сигнатурным анализом*.

Сигнатурный анализ разработан на основе двух применявшихся ранее способов контроля ошибок; первым является способ проверки логических узлов в цифровой системе, называемой *счетом переходов*, а вторым — способ контроля ошибок, называемый *циклическим избыточным контролем*. Сигнатурный анализ как бы моделирует первый способ, но действует аналогично второму способу.

Счет переходов и сигнатурный анализ предназначены для контроля аппаратных средств системы, и для их понимания необходимо рассмотреть виды сигналов в цифровых системах, включая и системы на базе компьютеров.

## 7.1. Природа цифровых сигналов

Периодический сигнал синхронизации синхронных и асинхронных цифровых схем характеризуется такими параметрами, как частота повторения и коэффициент заполнения. Коэффициент заполнения показывает, сколько времени сигнал находится в состоянии логической 1 по сравнению со временем нахождения его в состоянии логического 0. Если эти времена одинаковы, коэффициент заполнения равен 50 %. Частота повторения импульсов цифрового сигнала обратна интервалу времени между соседними положительными фронтами (или отрицательными фронтами) и характеризует только периодические сигналы.

Периодический цифровой сигнал на рис. 7.1 имеет частоту повторения импульсов, равную $1/T_C$. Временной интервал включенного состояния $T_1$ плюс временной интервал выключенного состояния $T_2$ дают период $T_C$. Сигнал характеризуется двумя параметрами — частотой повторения импульсов и коэффициентом заполнения, которые можно измерить с помощью обычного осциллографа. Единственным сигналом в цифровой вычислительной системе, который соответствует регулярному сигналу, приведенному на рис. 7.1, является сигнал синхронизации.

В цифровом компьютере или в любой сложной логи-

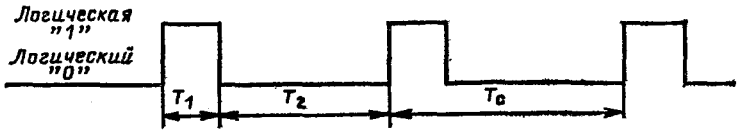



Рис. 7.1. Периодический цифровой сигнал

ческой схеме подавляющее большинство сигналов при наблюдении их на экране осциллографа выглядит почти случайным. Примерный вид сигнала показан на рис. 7.2, причем продолжительности уровней высокого и низкого напряжений на линии развертки могут значительно изменяться. Для такого сигнала невозможно определить такие характеристики, как частота повторения и коэффициент заполнения. Нестационарность сигналов означает, что для их идентификации нельзя применять статистические методы. Причины нерегулярной природы цифровых сигналов в сложных сетях следующие.

Любой отдельный узел в логической схеме, оперирующей данными в параллельном формате, содержит только часть информации, находящейся в системе в каждый момент времени. Эта часть изменяется почти случайным образом при изменении передаваемых внутри системы символов. Можно привести следующую наглядную аналогию данной ситуации: случайным образом возьмите по одной букве из каждого слова на этой странице и попробуйте восстановить по ним содержательный смысл.

На выполнение программ в компьютере влияют команды условных переходов, вызовы подпрограмм, запросы прерываний и операции прямого доступа к памяти. Их можно рассматривать как программно или аппаратно инициируемые асинхронные события, которые случайным образом воздействуют на логические сигналы. В обычных рабочих условиях охарактеризовать сигнал в отдельном узле невозможно из-за воздействия неожиданных событий.

Рис. 7.2. Непериодический цифровой сигнал

### 7.2. Счет переходов

Если узел стимулировать известным и периодическим тест-набором, то сигнал в узле можно считать псевдослучайной двоичной последовательностью, которая на коротких интервалах выглядит случайной, но повторяется при выборе большего масштаба времени.

Широко распространенный и простой способ контроля заключается в том, чтобы подсчитать переходы сигнала из одного состояния в другое и использовать полученное число в качестве идентификатора узла. Общее число изменений состояния стимулируемого узла может быть очень большим, поэтому необходим какой-то способ сжатия информации. Обязательным условием счета переходов является определенное «временно́е окно», в течение которого подсчитываются переходы в узле. В интервале «временно́го окна» выполняется тест-программа, которая стимулирует проверяемый узел и по возможности должна быть периодической для удобства воспроизведения и сравнения результатов нескольких измерений. В конкретной системе с определенными «временны́м окном» и тест-последовательностью результаты нескольких подсчетов переходов должны быть идентичными. Для применения метода счета переходов (СП) как способа поиска неисправностей необходимо измерить и задокументировать числа переходов в каждом узле. При возникновении неисправности исследователь выполняет тест-программы, измеряет числа переходов в подозреваемых узлах и сравнивает их с документированными значениями. Любые расхождения свидетельствуют о наличии неисправности, и с помощью систематической процедуры ее можно локализовать.

### 7.3. Вероятность успеха при счете переходов

Реализация счета переходов как способа контроля предполагает наличие прибора, который подсчитывает изменения состояний в узле между сигналами пуска и останова (они образуют «временно́е окно»). Как и любые другие электронные схемы, счетчик переходов подвержен ошибкам. Поскольку он служит показателем исправной работы проверяемой системы, нас интересует, насколько хорошо счетчик справляется со своей задачей. Длина тест-последовательности и число изменений состояния в

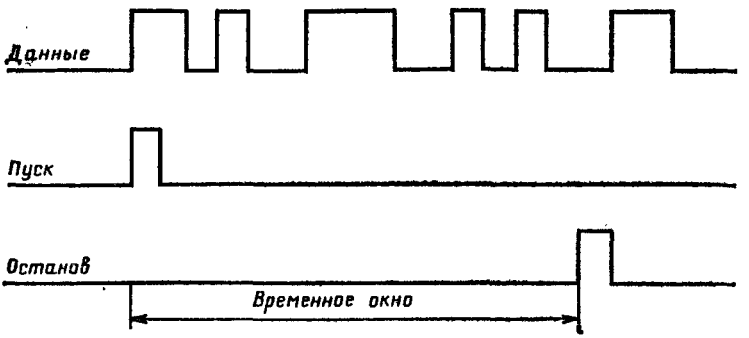



**Рис. 7.3.** Счет переходов

ней обычно неизвестны, поэтому мы можем предсказать только вероятность успеха при рассмотрении схемы переходов как общего способа контроля.

На рис. 7.3 показан простой иллюстративный пример, в котором временно́е окно содержит 9 переходов, исключая начальное изменение состояния на линии данных, совпадающее с нарастающим активным фронтом сигнала пуска.

Чтобы вычислить вероятность ошибки в приборе для счета переходов, предположим, что имеется $m$-битная двоичная последовательность, содержащая $r$ изменений состояния. В общем случае $r \neq 2m$, так как двоичный период определяется сигналом системной синхронизации и анализируемые данные представлены в формате без возврата к нулю, как показано на рис. 7.3.

Для двоичных сигналов имеется $2^m$ $m$-битных последовательностей, поэтому мы определим общее число $m$-битных двоичных последовательностей как

$$N_s = 2^m.$$

Счетчик переходов не может хранить правильную последовательность, которая может возникнуть в любом узле. Следовательно, при возникновении ошибки он не сможет соотнести ее с точным положением во времени. Порядок подсчета переходов оказывается несущественным, и возможное число способов подсчета $r$ переходов в $m$-битной последовательности определяется по правилу сочетаний:

$$N_r = \left(\frac{m}{r}\right) = \frac{m!}{r!\,(m-r)!}.$$

Предположим далее, что в системе возникает $N_t$ ошибок и $N_u$ ошибок остаются необнаруженными. Вероятность того, что счетчик переходов не обнаружит ошибку при условии возникновения некоторых ошибок,

$$P(\text{СП, нет}) = \frac{N_u}{N_t}.$$

Но

$$N_u = \sum_{r=0}^{m} P_{ur},$$

где $P_{ur}$ — вероятность необнаружения ошибок при наличии $r$ переходов. Очевидно,

$$P_{ur} = N_{ur} P_r,$$

где $N_{ur}$ — число необнаруженных ошибок при наличии $r$ переходов, а $P_r$ — вероятность подсчета $r$ переходов. Далее,

$$N_{ur} = N_r - N_c,$$

где $N_c$ — число правильных способов подсчета, которое может быть равно только 1 ($N_c = 1$). Кроме того,

$$P_r = \frac{N_r}{N_s}.$$

Теперь можно записать, что

$$N_u = \sum_{r=0}^{m} (N_r - 1) \frac{N_r}{N_s}$$

и

$$N_t = N_s - N_c = N_s - 1.$$

Следовательно,

$$P(\text{СП, нет}) = \sum_{r=0}^{m} \frac{(N_r - 1)(N_r/N_s)}{2^m - 1}.$$

Мы нашли вероятность того, что счетчик переходов не обнаружит ошибку, а тогда вероятность обнаружения ошибки

$$P(\text{СП}) = 1 - P(\text{СП, нет}).$$

Перейдя к процентам и подставив значение $N_r$, получим следующее выражение вероятности обнаружения

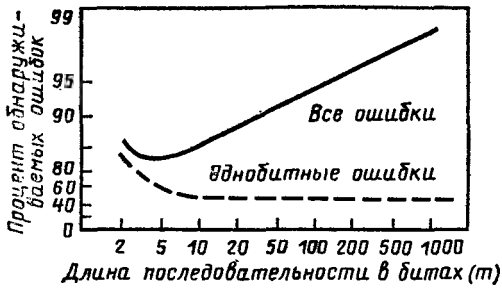



**Рис. 7.4.** Процент обнаруживаемых ошибок

ошибки счетчиком переходов в последовательности длины $m$:

$$P(\text{СП}\%) = 100 - \left(\frac{100}{2^m\,(2^m-1)}\right) \times$$

$$\times \sum_{r=0}^{m} \left[\frac{m!}{r!\,(m-r)!}\right]\left[\frac{m!}{r!\,(m-r)!} - 1\right] \approx$$

$$\approx 100\left(1 - \frac{1}{\sqrt{m\pi}}\right).$$

График этой зависимости при изменении $m$ приведен на рис. 7.4.

Полученное соотношение справедливо только в том случае, когда ошибки воздействуют на всю $m$-битную последовательность, а такая ситуация на практике встречается редко. Обычно ошибки влияют на несколько бит, а в худшем случае — всего на один бит.

Существуют $2^m$ возможных $m$-битных последовательностей, и изменение одного любого бита дает однобитную ошибку, поэтому в общем случае имеется $m \times 2^m$ однобитных ошибок.

Предположим, что начальное состояние счетчика переходов равно нулю. Если второй бит в $m$-битной последовательности содержит «1», то состояние первого бита может быть изменено и окажется необнаруженным. При этих условиях имеется $2^{m-1}$ способов завершения последовательности.

Из рис. 7.5 видно, что, если первый бит последовательности изменяется, когда второй бит содержит логическую

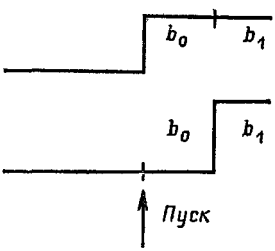



**Рис. 7.5.** Необнаруживаемое изменение состояния первого бита

«1», счетчик переходов зафиксирует только одно изменение состояния даже при наличии ошибки.

Второй возможный случай, когда счетчик переходов не обнаруживает однобитную ошибку, появляется, если переход происходит в середине такой комбинации, в которой внешние из 3 бит находятся в различных состояниях, а центральное состояние равно одному из соседних состояний. Этот случай характерен для двоичных наборов 001, 011, 100 и 110.

Из последовательностей «*A*» и «*B*» на рис. 7.6 видно, что, если центральный бит $b_i$ изменяет состояние, счетчик переходов фиксирует то же самое число изменений состояний и ошибка оказывается необнаруженной. Последовательности «*C*» и «*D*» также показывают, что однобитная ошибка в середине наборов 100 и 110 также не обнаруживается счетчиком переходов. В $m$-битной последовательности имеется $(m-2)\times 2^{m-1}$ таких средних ошибок.

С учетом двух источников необнаруживаемых ошибок получается $(m-1)\times 2^{m-1}$ последовательностей, содержащих однобитную ошибку, которую нельзя обнаружить счетом переходов. Поэтому вероятность необнаружения однобитной ошибки

$$P = \frac{(m-1)\times 2^{m-1}}{m\times 2^m} = \frac{m-1}{2m}.$$

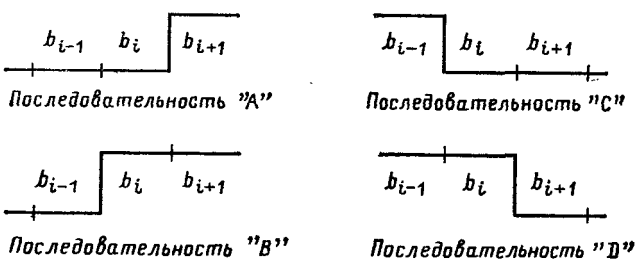



**Рис. 7.6.** Необнаруживаемые ошибки в середине потока

Эта вероятность при больших $m$ составляет около 50 %.

На практике разрядность счетчика составляет $n$ бит, т. е. меньше $m$. Допускается переполнение, и мерой правильной работы узла является остаток в регистре, получающийся по окончании счета. Переполнение вносит в измерение некоторую неоднозначность, и, в общем, в $n$-битном счетчике переходов наибольшее число обнаруживаемых ошибок равно $^1/_2{}^n$. Следовательно, переполнение увеличивает число ошибок, которое оказывается больше предсказанного, и более 50 % всех однобитных ошибок не обнаруживаются.

Из приведенных выкладок следует, что счет переходов как способ контроля ошибок оказывается не слишком надежным для наиболее вероятных ошибочных событий — однобитных ошибок. Однако, за исключением однобитных ошибок, счет переходов обеспечивает высокий процент обнаружения, особенно в случае длинной тест-последовательности. В течение длительного времени счет переходов применялся в автоматическом испытательном оборудовании как основной способ поиска неисправностей в логических схемах посредством контроля поведения узлов.

### 7.4. Коды циклического избыточного контроля

В вычислительной системе, где данные передаются, например с дискового накопителя в основную память, с высокой скоростью, необходимы средства контроля отсутствия ошибок при передаче. Простой контроль с битом четности оказывается недостаточным при передаче блоков данных и применим только в асинхронных последовательных линиях связи, в которых информация передается по символам. Для преодоления ограничений контроля по четности и обеспечения обнаружения одно- и многобитных ошибок в быстродействующих синхронных линиях связи были разработаны коды циклического избыточного контроля.

Если взять 8-битное двоичное слово, его можно рассматривать как коэффициент полинома степени 7. Двоичный набор $B_7B_6B_5B_4B_3B_2B_1B_0$ есть сокращенное представление двоичного полинома:

$$B_7 \cdot 2^7 + B_6 \cdot 2^6 + B_5 \cdot 2^5 + B_4 \cdot 2^4 + B_3 \cdot 2^3 + \\ + B_2 \cdot 2^2 + B_1 \cdot 2^1 + B_0 \cdot 2^0.$$

Например, двоичное слово 11000001 можно представить в виде

$$1 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^0.$$

Для простоты обозначим переменную в полиноме вместо конкретного значения 2 символом $X$. Тогда то же самое слово с фиктивной переменной $X$ принимает вид

$$A(X) = 1 \cdot X^7 + 1 \cdot X^6 + 0 \cdot X^5 + 0 \cdot X^4 + 0 \cdot X^3 + \\ + 0 \cdot X^2 + 0 \cdot X^1 + 1 \cdot X^0.$$

Данный полином можно сократить до выражения

$$A(X) = X^7 + X^6 + X^0 = X^7 + X^6 + 1.$$

Для понимания принципа циклического избыточного контроля (ЦИК) предположим, что имеются двоичный поток $B(X)$ и порождающий полином $G(X)$. Если разделить полином $B(X)$ на $G(X)$, получится частное $Q(X)$ и остаток $R(X)$:

$$B(X) = G(X)\,Q(X) + R(X).$$

Прибавляя $-R(X)$ к обеим частям уравнения, получим

$$B(X) - R(X) = G(X)\,Q(X).$$

При способе ЦИК передаваемый двоичный поток делится на порождающий полином и получающийся остаток добавляется в передаваемый двоичный поток. В приемнике входной двоичный поток и остаток делятся на тот же самый полином, что и в передатчике, с образованием остатка. В результате деления на приемной стороне линии связи остаток должен быть равен нулю, так как $B(X)+R(X)$ точно делится на $G(X)$. При получении нулевого остатка на приемной стороне обнаруживаемых ошибок при передаче не возникло.

При способе ЦИК наиболее широко применяется 16-битный полином вида

$$\text{ЦИК-16} = X^{16} + X^{15} + X^2 + 1.$$

Для реализации циклического избыточного контроля применяется 16-битный регистр сдвига, на вход которого подается контролируемый двоичный набор $B(X)$, а из соответствующих разрядов регистра сдвига берутся сигналы обратной связи. Входной сигнал и все сигналы обратной связи суммируются по mod 2 (операция исклю-

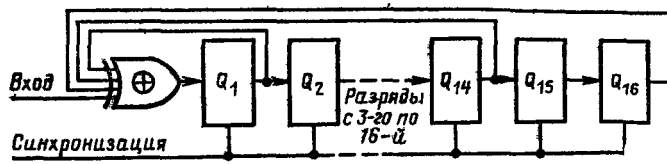



**Рис. 7.7.** Регистр сдвига с линейной обратной связью

чающего ИЛИ), в результате чего получается линейная последовательностная схема, так как сумматор по mod 2 придает один и тот же вес каждому входному биту. Полином называется характеристическим полиномом генератора, а фактические точки отводов обратной связи находятся из его инверсии. Инверсное (или обратное) выражение находится путем вычитания каждого члена в характеристическом полиноме из $X^{16}$ в случае кода ЦИК-16, что дает выражение обратной связи $X^{16}+X^{14}+$ $+X^{1}+1$. Следовательно, в элемент исключающего ИЛИ подаются входной сигнал и отводы от первого, четырнадцатого и шестнадцатого разрядов регистра сдвига (рис. 7.7).

Регистр сдвига синхронизируется специальными сигналами, и входной поток данных делится на характеристический полином $X^{16}+X^{15}+X^{2}+1$. По окончании двоичного потока остаток представлен состояниями 16 триггеров; остаток выдвигается из регистра и добавляется к передаваемому потоку данных. Ранее было показано, что остаток вычитается из двоичного потока $[B(X)-R(X)]$, но в арифметике по mod 2 операции сложения и вычитания дают одинаковые результаты, поэтому передача $B(X)+R(X)$ производит то же самое действие.

Порождающий полином ЦИК-16 имеет четное число членов, и представим в виде ЦИК-16$=(X+1)(X^{15}+X+$ $+1)$. Множитель $(X+1)$ включен намеренно, чтобы все необнаруживаемые ошибки имели четный паритет. Чтобы скрыть ошибку, должно возникнуть четное число ошибочных бит, которые обычно группируются. Однако циклический избыточный контроль всегда обнаруживает однобитные ошибки во входном потоке. Таким образом, коды ЦИК представляют собой способ обнаружения в потоке данных всех однобитных ошибок и высокий процент обнаружения всех многобитных ошибок. В связи с этим они широко применяются в схемах быстродействующих циф-

ровых передач, в которых передаются не отдельные символы, а целые блоки данных.

Счет переходов и коды ЦИК предназначены для обнаружения ошибок в длинных двоичных потоках данных. Первый способ обеспечивает высокую вероятность успеха для многобитных ошибок, а коды ЦИК всегда обнаруживают однобитные ошибки и имеют приемлемую эффективность для многобитных ошибок. Было бы хорошо иметь метод тестирования, который имел бы в своей основе анализ поведения отдельных узлов в логической схеме, реализованный как счет переходов, но имел бы такую же эффективность, как и коды ЦИК. Это объединение характеристик двух способов реализовано в сигнатурном анализе, разработанном фирмой Hewlett-Packard.

### 7.5. Сигнатурный анализ

В циклическом избыточном контроле входной двоичный набор подается в линейную последовательностную схему, которая осуществляет деление двоичного потока на некоторый характеристический полином, и в регистре сдвига образуется остаток от деления. Обычно остаток добавляется к передаваемому двоичному потоку в качестве кода, обнаруживающего ошибки. Если же вместо добавления остатка к двоичному потоку вывести его на индикацию, это значение будет уникальным для входного двоичного набора. Имея запоминающие элементы, схема учитывает все прошлые и текущие события и может обрабатывать очень длинные потоки данных. Уникальный остаток для конкретного входа служит как бы «отпечатками пальцев» этого набора и может использоваться для его идентификации. Зависимость остатка от входного двоичного потока привела к термину «сигнатура» (т. е. подпись). Опираясь на принцип «временно́го окна» и используя импульсы пуска и останова и сигналы синхронизации от проверяемой системы, в узел логической схемы можно подать периодический набор. Этот набор подается на вход линейной последовательностной схемы, и при восприятии сигнала останова в регистре сдвига окажется «сигнатура» данного узла и конкретного тест-набора. Собственно, мы изложили принцип сигнатурного анализа. Фактическое значение сигнатуры не существенно, но оно должно быть одним и тем же для данного узла, когда он стимулируется одним и тем же

тест-набором и когда применяются те же самые сигналы пуска, останова и синхронизации.

Для получения наилучших результатов при применении способа регистра сдвига требуется последовательность максимальной длины, что приводит к широкому классу схем, называемых *генераторами псевдослучайной последовательности.* В 16-разрядном регистре сдвига имеется 2048 способов реализации отводов обратной связи, удовлетворяющих данному критерию. В полиноме ЦИК-16 применяется четное число входов, что приводит к группированию ошибок, а при тестировании узлов предпочтителен метод, который максимально распределяет ошибки. По этой же причине отводы не рекомендуется делать через 4 или 8 разрядов, так как они соответствуют наиболее вероятным размерам слов в микропроцессорах. Фирма Hewlett-Packard остановилась на нечетном числе входов, применив неприводимое выражение обратной связи $X^{16}+X^{12}+X^{9}+X^{7}+1$, которое соответствует характеристическому полиному $X^{16}+X^{9}+X^{7}+X^{4}+1$. Напомним, что мы хотим получить прибор широкого назначения для тестирования цифровых систем; имеются и другие характеристические выражения, которые удовлетворяют критерию, но было выбрано именно это.

По существу, получается портативный прибор, содержащий 16-разрядный регистр сдвига с сумматором по mod 2 на входе. С помощью сигналов пуска и останова входной сигнал подается в схему, а сигнал синхронизации от проверяемой системы сдвигает данные в регистре. По окончании интервала измерения осуществляется индикация содержимого регистра как характеристической сигнатуры проверяемого узла.

На рис. 7.8 показана основная часть сигнатурного анализатора, в котором применяется 16-разрядный ре-

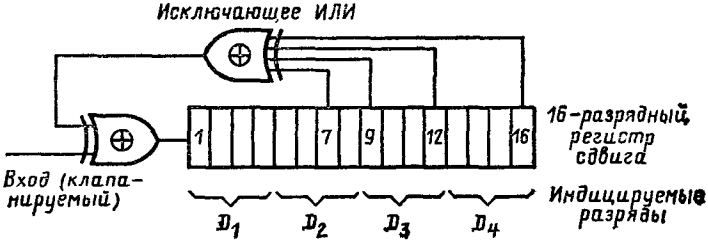



**Рис. 7.8.** Генератор псевдослучайной двоичной последовательности, применяемый в сигнатурном анализе

гистр сдвига. Входной двоичный поток от проверяемого узла с помощью сигналов пуска и останова подается в регистр, который синхронизируется сигналом от проверяемой системы. После прекращения двоичного потока остаток, находящийся в регистре, выводится на индикаторы в виде четырех 16-ричных символов, которые и представляют собой «сигнатуру» проверяемого узла.

Фирма Hewlett-Packard заменила стандартный набор 16-ричных символов, чтобы избежать путаницы между цифрой 6 и буквой *b*, и приняла следующие символы:

| ДВОИЧНЫЙ КОД | ИНДИКАЦИЯ | ОБЫЧНЫЙ 16-РИЧ. СИМВОЛ |
|---|---|---|
| 0000 | 0 | 0 |
| 0001 | 1 | 1 |
| 0010 | 2 | 2 |
| 0011 | 3 | 3 |
| 0100 | 4 | 4 |
| 0101 | 5 | 5 |
| 0110 | 6 | 6 |
| 0111 | 7 | 7 |
| 1000 | 8 | 8 |
| 1001 | 9 | 9 |
| 1010 | A | A |
| 1011 | C | B |
| 1100 | F | C |
| 1101 | H | D |
| 1110 | P | E |
| 1111 | U | F |

Большинство изготовителей сигнатурных анализаторов пользуется таким же кодированием индицируемых данных, что и фирма Hewlett-Packard.

Такое понятие, как почти правильная сигнатура, не имеет смысла; индицируемый код 006A совершенно не связан с кодом 006C. Сигнатура может быть только правильной или неправильной.

### 7.6. Вероятность успеха в сигнатурном анализе

Прежде чем вычислять вероятность успешного обнаружения ошибки с помощью сигнатурного анализа, рассмотрим работу более простой последовательной схемы. Схема состоит из 4-разрядного регистра сдвига и

сумматора по mod 2 и часто называется генератором последовательности максимальной длины или генератором псевдослучайной последовательности (ГПСП).

Работу регистра сдвига можно описать с помощью оператора задержки $D$, определяемого соотношением $X(t)=DX(t-1)$. Умножение на $D$ эквивалентно задержке данных на одну единицу времени. Мы рассматриваем только синхронные системы, поэтому все события всегда происходят через регулярные временные интервалы. В вычислительной системе этот интервал определяется системной синхронизацией.

В схеме на рис. 7.9 входные данные в регистр представляют собой сумму по mod 2 собственно входных данных и кодов из регистра, взятых через один и через четыре периода синхронизации. Выражение обратной связи можно записать в виде $D^4X(t)+DX(t)+X(t)$, или, проще, в виде $X^4+X+1$. Ввод потока данных в ГПСП эквивалентен делению потока данных на характеристический полином генератора, который для этой схемы описывается выражением $X^4+X^3+1$. Это можно доказать, взяв определенный вход, например 1 000 000 000 000 000, и выполнив деление вручную. Можно убедиться в правильности результата, составив таблицу последовательности состояний по мере поступления данных в регистр. Если предположение справедливо, при обоих способах получится один и тот же результат.

Пусть в начальном состоянии все разряды ГПСП находятся в нулевом состоянии. (Такое состояние недопустимо в автономном ГПСП, который может проходить через $2^N-1$ состояний и не может иметь на всех выходах нули.) Если представить себе схему на рис. 7.9 без внешней входной линии, то при нулевом состоянии всех разрядов на выходе элемента исключающего ИЛИ

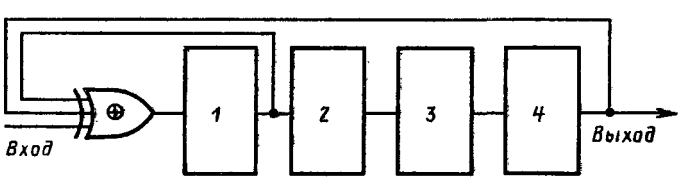



**Рис. 7.9.** Четырехразрядная линейная последовательностная схема

также будет нуль, и при действии сигналов синхронизации регистр просто сдвинет нули. Схема блокируется в таком состоянии и не может выйти из него. В схеме собственно генератора псевдослучайной последовательности необходимы средства первоначальной установки хотя бы одного разряда в логическую 1, чтобы осуществить автоматический запуск генератора. Однако в нашем случае нулевое начальное состояние допустимо, так как на внутренние последовательности регистра будет воздействовать входная последовательность. Если входная последовательность состоит из логических 0, регистр будет оставаться в нулевом состоянии.

| ТАКТ | $Q_4$ $Q_3$ $Q_2$ $Q_1$ | СЛЕДУЮЩИЙ ВХОД В РЕГИСТР | СЛЕДУЮЩИЙ ВХОД |
|---|---|---|---|
| 0 | 0000 | 1 | 1 |
| 1 | 0001 | 1 | 0 |
| 2 | 0011 | 1 | 0 |
| 3 | 0111 | 1 | 0 |
| 4 | 1111 | 0 | 0 |
| 5 | 1110 | 1 | 0 |
| 6 | 1101 | 0 | 0 |
| 7 | 1010 | 1 | 0 |
| 8 | 0101 | 1 | 0 |
| 9 | 1011 | 0 | 0 |
| 10 | 0110 | 0 | 0 |
| 11 | 1100 | 1 | 0 |
| 12 | 1001 | 0 | 0 |
| 13 | 0010 | 0 | 0 |
| 14 | 0100 | 0 | 0 |
| 15 | 1000 | 1 | 0 |

Регистр синхронизируется на 16 входных бит, а его начальное состояние предполагается нулевым. Оно показано в таблице как такт 0; выходы всех 4 триггеров $Q_4$—$Q_1$ содержат логические 0, а первый бит потока данных (логическая 1) является следующим входом. Сумматор по mod 2 на входе суммирует входной бит и состояния $Q_4$ и $Q_1$. Если сумма — нечетное число, на выходе сумматора появляется логическая 1. В такте 0 сложение трех входов дает нечетное число, поэтому на выходе сумматора появляется логическая 1; она показана в таблице как следующий вход в регистр.

Выходной двоичный поток из разряда $Q_4$ 0 000 111 101 011 001 получается при считывании столбца $Q_4$ сверху вниз.

Характеристический полином регистра $X^4+X^3+1$ расширяется до полного представления $1X^4+1X^3+0X^2+$ $+0X^2+0X^1+1X^0$. Это выражение соответствует двоичному числу 11001. Если взять входной поток данных и разделить его на 11011, получим следующее (напомним, что результаты сложения и вычитания по mod 2 двоичных переменных одинаковы и переносы отсутствуют):

```
 1        1  0  0  1          0 000 111 101 011 001
 ↑        ↑  ↑  ↑  ↑          ---------------------
Входной бит Q4 Q3 Q2 Q1       1 000 000 000 000 000
                              1 100 1
                              -------
                                100 10
                                110 01
                                -------
                                 10 110
                                 11 001
                                 -------
                                  1 111 0
                                  1 100 1
                                  -------
                                     11 100
                                     11 001
                                     -------
                                        101 00
                                        110 01
                                        -------
                                         11 010
                                         11 001
                                         -------
                                            11 000
                                            11 001
                                            -------
                                                1
```

Полученный результат 0 000 111 101 011 001 совпадает с найденным через двоичные последовательности.

Обратная связь через элемент исключающего ИЛИ в регистре сдвига дает равные веса всем входным битам.

Полином называется линейным, если $P(X+Y)=$ $=P(X)+P(Y)$. В рассмотренном ниже примере в один и тот же ГПСП подаются три различных двоичных потока $X$, $Y$, и $X+Y$.

Результатом работы ГПСП для каждого входа являются частное $Q$ и остаток $R$. Отметим, что выходные последовательности формируются как функция исключающего ИЛИ, т. е. $Q(X+Y)=Q(X)\oplus Q(Y)$. Здесь $Y$ представляет собой одиночный импульсный бит, задер-

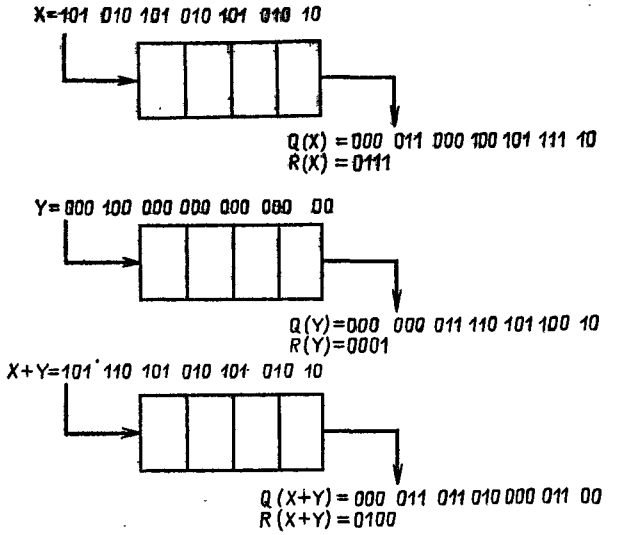



Рис. 7.10. Иллюстрация линейности регистра сдвига с обратной связью

жанный во времени от начала основной последовательности $X$, и единственное различие между $X$ и $(X+Y)$ заключается в одном бите $Y$. После первых 7 бит выходные двоичные потоки от $X$ и $(X+Y)$ не имеют сходства друг с другом, и в 20-битной входной последовательности остатки $R(X)$ и $R(X+Y)$ совершенно не похожи друг на друга: $R(X) = 0111$ и $R(X+Y) = 0100$ (рис. 7.10).

Если считать $X$ действительной входной последовательностью, а $Y$ — битом ошибки, то результирующий вход в ГПСП будет $(X+Y)$, так как к линейной схеме применима теорема суперпозиции. Из приведенного выше следует, что даже однобитная ошибка обнаруживается как изменение правильной выходной последовательности и, более того, выходная последовательность остается ошибочной независимо от длины входной последовательности. Остаток в регистре после конечного числа входных сигналов отличается от остатка, полученного для правильной последовательности. В примере на рис. 7.10 остаток $R(X+Y)$ отличается от правильного остатка $R(X)$, и действие ошибки сохраняется, несмотря

на то что бит ошибки исчез много периодов синхронизации назад.

Остаток зависит от каждого бита, поступившего в регистр, и является суммарной характеристикой ГПСП и входного потока данных; именно поэтому остаток называется СИГНАТУРОЙ.

### 7.6.1. Обнаружение ошибки при сигнатурном анализе

Генератор псевдослучайной последовательности (ГПСП) можно использовать для обнаружения однобитной ошибки в потоке данных любой длины.

Предположим, что $X$ — поток данных длиной $m$ бит, $P$ — $n$-битный ГПСП и $P^{-1}$ — такая его инверсия, что $P \cdot P^{-1} = 1$. Обозначим через $Q$ частное, а $R$ — остаток. Имеем

$$P(X) = Q(X)\,2^n + R(X). \qquad (1)$$

Возьмем еще одну $m$-битную последовательности, не совпадающую с $X$ и, следовательно, отличающуюся от нее на $m$-битную последовательность ошибки $E$, так что

$$Y = X + E.$$

Очевидно,

$$P(Y) = Q(Y)\,2^n + R(Y).$$

Получаем

$$P(X + E) = Q(X + E)\,2^n + R(X + E).$$

Так как все операторы линейны, будем иметь

$$P(X) + P(E) = Q(X)\,2^n + Q(E)\,2^n + R(X) + R(E).$$

Вычитание (или сложение) по mod 2 выражения (1) дает

$$P(E) = Q(E)\,2^n + R(E). \qquad (2)$$

Чтобы последовательность $Y$ содержала необнаруживаемые ошибки, должно быть

$$R(Y) = R(X).$$

Отсюда получаем

$$R(Y) = R(X + E) = R(X) + R(E) = R(X),$$

и, следовательно, $R(E) = 0$. Подставим это значение в

выражение (2):

$$P(E) = Q(E)\,2^n,$$

и тогда все необнаруживаемые ошибки определяются из соотношения

$$E = P^{-1}\,Q(E)\,2^n. \qquad (3)$$

В случае однобитной ошибки

$$E = D^a(1),$$

где $D$ — оператор задержки; $a$ — период задержки и 1 есть импульсная последовательность 100000... Подставим значение $E$ в выражение (3):

$$D^a(1) = P^{-1}\,Q\left(D^a(1)\right)2^n.$$

Оператор $D$ коммутативен с другими линейными операторами, поэтому

$$D^a(1) = D^a P^{-1}\,Q(1)\,2^n;$$

$$1 = P^{-1}\,Q(1)\,2^n;$$

$$P(1) = Q(1)2^n.$$

Однако по первоначальным предположениям

$$P(1) = Q(1)\,2^n + R(1).$$

откуда с помощью вычитания получаем

$$R(1) = 0.$$

В предыдущем примере было показано, что остаток от однобитной ошибки не равен нулю. Следовательно, $E \neq D^a(1)$, и множество необнаруживаемых ошибок не включает никакие однобитные ошибки. Другими словами, при использовании ГПСП обнаруживаются ВСЕ однобитные ошибки.

Так как $X$, $Y$ и $E$ являются $m$-битными последовательностями, то $Q2^n$ должен быть $m$-битной последовательностью, содержащей $n$ заключительных нулей. Поэтому $Q$ содержит $m-n$ бит и должно быть $2^{m-n}$ последовательностей, которые отображаются в тот же самый остаток, что и правильная последовательность. Из этих последовательностей $2^{m-n}-1$ оказываются ошибочными последовательностями, которые дают такой же остаток, как и правильная последовательность. Из $m$ бит можно образовать $2^m$ последовательностей, и только од-

на из них будет правильной. Тогда вероятность необнаружения ошибки с применением ГПСП определится выражением

$$P(\text{ГПСП, нет}) = \frac{\text{Необнаруживаемые ошибки}}{\text{Общее число ошибок}}.$$

Отсюда получаем

$$P(\text{ГПСП, нет}) = \frac{2^{m-n} - 1}{2^m}.$$

Для длинных последовательностей (большое значение $m$) вероятность стремится к пределу:

$$P(\text{ГПСП, нет}) = 1/2^n.$$

Регистр сдвига с обратной связью, имеющий $n$ разрядов, обнаружит все ошибки в потоках данных длиной $n$ и менее бит, так как в регистре сохранится вся последовательность и $R(X) = P(X)$.

Для потоков данных длиной более $n$ бит вероятность обнаружения ошибки близка к 1 даже в генераторах с умеренным числом разрядов. Необнаруживаемые ошибки предсказуемы, и их можно получить, если взять все $m$-битные последовательности, имеющие $n$ последних нулей, и воздействовать на них инверсией полинома $n$-битного ГПСП, т. е.

$$E = P^{-1}(Q2^n).$$

В 16-битном ГПСП вероятность необнаружения ошибки равна:

$$E = 1/2^{16} = 0{,}00001526,$$

а вероятность обнаружения всех ошибок составляет

$$E = 1 - 0{,}00001526 = 0{,}999985.$$

Таким образом, с использованием 16-битного ГПСП все ошибки обнаруживаются с вероятностью 99,9985 %.

### 7.7. Простой сигнатурный анализатор

Дешевый сигнатурный анализатор можно собрать из стандартных логических ИС. Основу его составляет 16-разрядный регистр сдвига с элементом исключающего ИЛИ в цепи обратной связи. Первоначально регистр переводится в нулевое состояние, которое является запре-

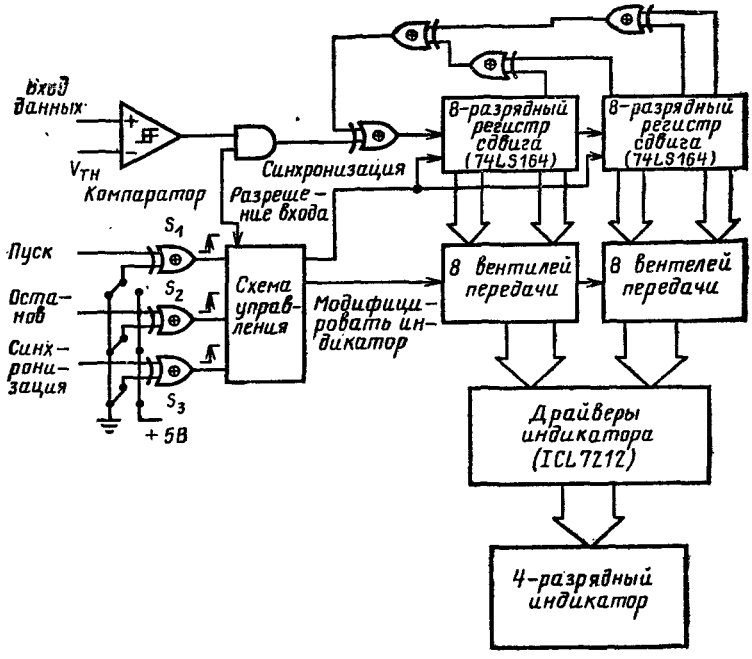



**Рис. 7.11.** Простой сигнатурный анализатор

щенным состоянием для автономного ГПСП, но из которого его можно вывести любым битом с логической 1 во входном потоке данных. Сигналы пуска, останова и синхронизации подаются из проверяемой системы вместе с сигналом от проверяемого узла. Сигнал пуска разрешает прохождение сигналов синхронизации в регистр сдвига, поэтому данные можно синхронно сдвигать в регистре. Сигналом пуска можно также клапанировать входной поток данных. Сигнал останова прекращает синхронизацию регистра сдвига и блокирует входные данные. Сигналом останова остаток из регистра сдвига передается в драйверы индикатора.

Структурная схема простого сигнатурного анализатора приведена на рис. 7.11.

Схема на рис. 7.11 иллюстрирует несколько интересных особенностей. 16-разрядный регистр сдвига реализован на двух микросхемах 74LS164, а цепь обратной связи построена на двухвходовых элементах исключающего

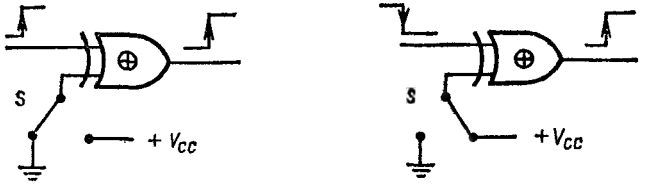



**Рис. 7.12.** Элемент исключающего ИЛИ изменяет действующий фронт входного сигнала

ИЛИ 74LS86. Входной поток данных для улучшения формы сигналов подается на компаратор, который сравнивает уровень входного сигнала с пороговым напряжением $V_{TH}$.

В приборе широкого назначения необходимо иметь возможность устанавливать для сигналов пуска, останова и синхронизации любой активный фронт — нарастающий или спадающий. Например, в одном тесте нужен сигнал пуска с активным нарастающим фронтом, а в другом тесте он должен запускать операции спадающим фронтом. Возможность задания активного фронта сигнала обеспечивается входными элементами исключающего ИЛИ, через которые сигналы проходят в схему управления, как показано на рис. 7.12. В случае сигнала пуска переключатель $S_1$ подсоединяется к земле или к источнику питания $V_{cc}$. Когда ключ замкнут на землю, на выходе элемента исключающего ИЛИ повторяется входной сигнал пуска. Если же $S_1$ подключен к $V_{cc}$, сигнал на выходе представляет собой инверсию входного сигнала. Предположим, что для инициирования схемы управления всегда требуется нарастающий фронт сигнала. Тогда для удовлетворения этого требования с помощью $S_1$ можно выбрать либо нарастающий фронт сигнала ($S_1$ подключен к земле), либо спадающий фронт сигнала ($S_1$ подключен к $V_{cc}$). Рисунок 7.12 показывает, как можно выбрать любой фронт сигнала для инициирования и прекращения работы схемы управления.

Остаток, сформированный в регистре сдвига, при восприятии сигнала останова передается в регистр-защелку. Информация из этого регистра индицируется как «сигнатура» проверяемого узла. Для управления 4 светодиодными индикаторами можно использовать 4-разрядный драйвер ICL7212 фирмы Intersil. Отметим, что информация индицируется в стандартном 16-ричном

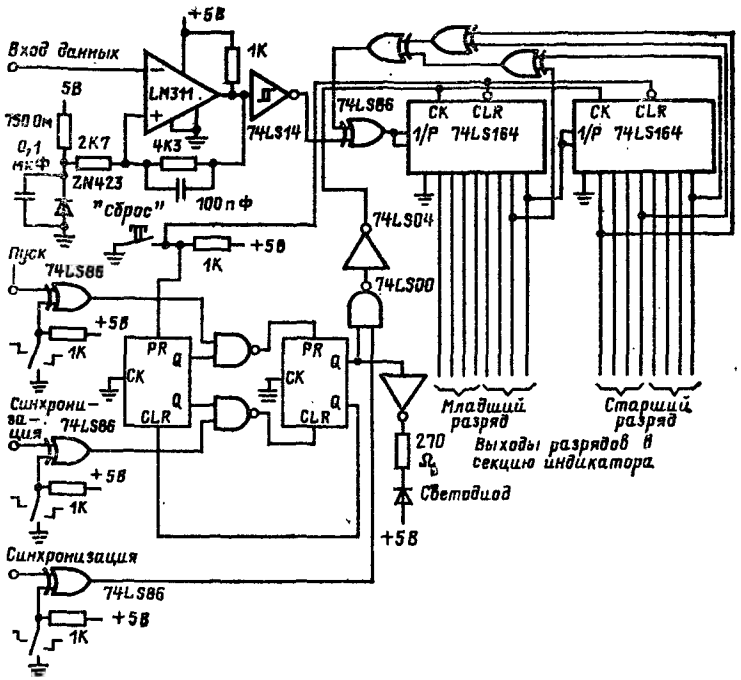



Рис. 7.13. Секция анализа простого сигнатурного анализатора

формате, а не в специальном формате фирмы Hewlett-Packard. Наличие однозначного соответствия между обоими форматами, к которому легко привыкнуть, устраняет этот недостаток.

На рис. 7.13 показана секция формирования сигнатур простого сигнатурного анализатора. Линия входных данных подается на гистерезисный коммутатор, выполненный на основе компаратора LM311. Пороговые уровни компаратора составляют 0,8 и 2,0 В, что соответствует стандартным ТТЛ-уровням.

До начала работы прибора оператор вручную сбрасывает систему, при этом устанавливается начальное состояние схемы управления, а 16-разрядный регистр сдвига переводится в нулевое состояние. После сброса сигнал пуска разрешает прохождение сигнала синхронизации в регистр сдвига, который синхронно сдвигает данные до появления сигнала останова. Светодиод в

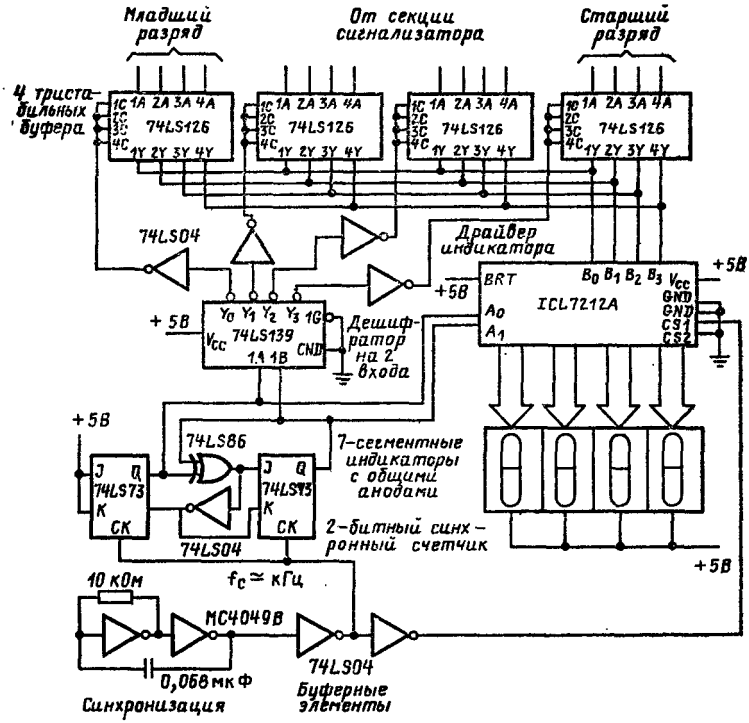



**Рис. 7.14.** Секция индикации простого сигнатуриого анализатора

схеме управления показывает, что прибор формирует сигнатуру.

Схема индикации сигнатуры, зарегистрированной в анализаторе, приведена на рис. 7.14. Секции индикатора и анализатора работают асинхронно. Выходы регистра сдвига подаются на тристабильные буферы 74LS126. Каждый из них последовательно выбирается дешифратором, а выходы буферов передаются на выходную шину и в драйвер индикатора ICL7212. Драйвер управляет четырьмя светодиодными индикаторами с общим анодом в немультиплексном режиме. Основные сигналы синхронизации, которые управляют секцией индикатора, формируются с помощью КМОП-инвертора и имеют частоту около 1 кГц. Сигналы синхронизации подаются в двухбитный синхронный счетчик, который обеспечивает 4 состояния, необходимых для управления буферами.

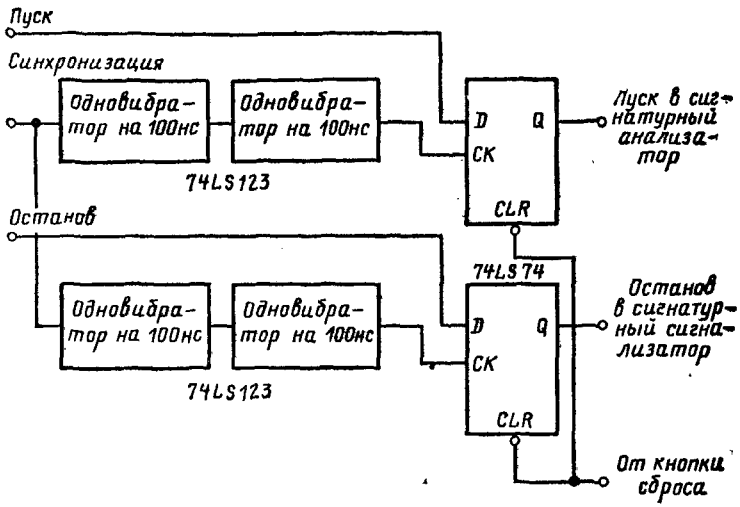



Рис. 7.15. Схема задержки сигналов пуска и останова

Данные индицируются в стандартном 16-ричном формате, причем цифра 6 имеет «хвостик», позволяющий отличить ее от малой буквы *b*. В некоторых системах потребовалось ввести перекос в сигналы пуска и останова, чтобы нестабильные фронты сигналов не запускали регистрацию данных. Эта проблема решается с помощью схемы, показанной на рис. 7.15. Она включается между выходами элементов исключающего ИЛИ секции анализатора и входами в схему управления. Схема обеспечивает задержку 100 нс между входными сигналами пуска и останова и сигналами, осуществляющими запуск анализатора. Такой задержки достаточно для стабилизации сигналов, которые начинают и оканчивают регистрацию данных.

Описанный нами анализатор относительно прост, но является вполне работоспособным прибором и стоит примерно в 10 раз дешевле анализаторов, выпускаемых промышленностью.

### 7.8. Тестирование в режиме свободного счета с использованием сигнатурного анализатора

Сигнатурный анализатор можно использовать для проверки ядра системы, образуя сигнатуры узлов в ре-

жиме свободного счета. Данный режим работы рассмотрен в § 2.5.

При проведении любого теста с применением сигнатурного анализа необходимо решить, какие сигналы от проверяемой системы следует использовать в качестве сигналов пуска, останова и синхронизации. В промышленных сигнатурных анализаторах зонд для касания узла имеет логический пробник, который дает визуальную индикацию активности. Конечно, индикатор пробника не дает возможности определить природу действий в узле, но он показывает наличие или отсутствие сигналов в проверяемом узле.

Предположим, что исследуется система с 8-битным микропроцессором, имеющим 16-битную шину адреса. В цикле свободного счета на шине адреса возникают все двоичные наборы, которые циклически повторяются. Благодаря периодичности такой тип активности идеально подходит для сигнатурного анализа. На старшей линии $A_{15}$ шины адреса действует низкий уровень для одной половины всех адресов и высокий уровень для другой половины. Следовательно, между соседними нарастающими фронтами сигнала на линии $A_{15}$ находится один полный цикл шины адреса. Сигнал с этой линии можно использовать как сигналы пуска и останова анализатора. Первый из них осуществляет запуск, а второй — останов. Остаток, образованный в регистре сдвига между этими событиями, подается на индикатор в качестве сигнатуры проверяемого узла. В режиме свободного счета все команды осуществляют считывание из памяти, и сигналы для анализатора можно взять с линии $\overline{\text{READ}}$. Анализатор настраивается на нарастающие фронты входов пуска, останова и синхронизации, поэтому данные синхронно проходят через регистр сдвига по заднему фронту сигнала $\overline{\text{READ}}$ в течение одного полного цикла шины адреса. Необходимые подключения показаны на рис. 7.16.

До взятия сигнатур от узлов в системе сам сигнатурный анализатор и подключения входных сигналов контролируются по сигнатурам земли и питания $V_{cc}$. Регистр сдвига в анализаторе инициализируется на нуль до регистрации любых данных. Когда пробник касается земли, вход данных всегда находится в состоянии логического 0, которое не изменяет начального состояния реги-

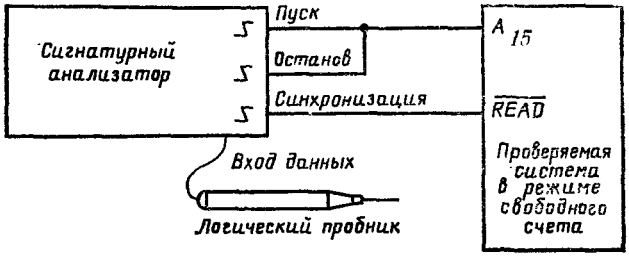



**Рис. 7.16.** Получение сигнатур в режиме свободного счета

стра сдвига. По окончании цикла регистрации данных остаток в регистре сдвига будет нулевым. Это состояние может изменить только входной сигнал логической 1, которого, очевидно, не может быть при контроле сигнатуры земли. Следовательно, земля всегда дает сигнатуру 0000, которую можно считать ее «характеристической сигнатурой». Однако положительное питание $V_{cc}$ всегда воспринимается как состояние логической 1, которое изменяет начальное состояние регистра сдвига. Остаток, образующийся по окончании регистрации данных, зависит от числа состояний синхронизации между сигналами пуска и останова и будет различным при изменении запускающих сигналов. При конкретном подключении сигналов пуска, останова и синхронизации сигнатура $V_{cc}$ будет одной и той же, поэтому ее называют «характеристической сигнатурой» для данного подключения входов. Но, разумеется, она будет получаться различной при других подключениях управляющих входов и (или) выборе других активных фронтов. Когда от проверяемого узла получается такая же сигнатура, как и от $V_{cc}$, может оказаться, что из-за отказа он закорочен на шину питания $V_{cc}$. Однако иногда и от исправных узлов получается такая же сигнатура, как и характеристическая сигнатура $V_{cc}$. Проще всего различать эти две ситуации по индикатору логического пробника — в исправном узле он вспыхивает, показывая наличие сигналов в узле. Если же индикатор не вспыхивает, следует предположить наличие отказа.

Для любой системы, рассчитанной на сигнатурный анализ, должен существовать документ, в котором приведены сигнатуры всех узлов. В режиме свободного счета, хотя он и сканирует все адресное пространство,

будут разрешены не все микросхемы, так как микропроцессор выполняет только операции считывания из памяти. Например, входной порт не разрешается, и получить от него значимые сигнатуры невозможно. Шина данных отключена от процессора и также не дает значимых сигнатур (за исключением некоторых специальных условий). Набор сигнатур берется от заведомо исправной системы и документируется. В начале списка сигнатур показываются подключения входов пуска, останова и синхронизации, а также их активные фронты (нарастающий или спадающий). Кроме того, здесь же даются характеристические сигнатуры для указанных подключений входов. Эта информация необходима для настройки анализатора и контроля подключений по сигнатурам, полученным от $V_{cc}$ и земли. После этого берутся сигнатуры от узлов, и результаты сравниваются с приведенными в документе сигнатурами. Пример таблицы сигнатур для режима свободного счета приведен на рис. 7.17.

Таблица представляет собой только небольшую часть полного перечня сигнатур, которые берутся для каждой микросхемы в системе. В начале таблицы показаны положения переключателей и используемые от системы сигналы.

В данном примере характеристическая сигнатура $V_{cc}$ равна 0001; ее необходимо проверить до взятия от системы остальных сигнатур. В режиме свободного счета сигналы на шине данных бессмысленны, что показано в таблице в виде набора состояний «безразлично» ($X$). Однако шина адреса проверяется, поэтому приведены сигнатуры для всех линий шины адреса.

Далее в документе находятся диаграммы разводки выводов всех микросхем, и у каждого вывода показана его сигнатура. Земля всегда имеет характеристическую сигнатуру 0000, которая приводится как GND. Чтобы показать, что сигнатура 0000 допустима для вывода и «отличается» от сигнатуры земли, после сигнатуры находится буква $B$. Она показывает, что светодиод, находящийся в зонде логического пробника, при взятии сигнатуры будет вспыхивать. Примером служит сигнатура у вывода *18* микросхемы ИС2. В режиме свободного счета сигнатуры на многих выводах ИС не имеют смысла и показываются на диаграммах в виде $X$ (см. пример у вывода *3* ИС4). Еще одна часто встречающаяся ситуа-

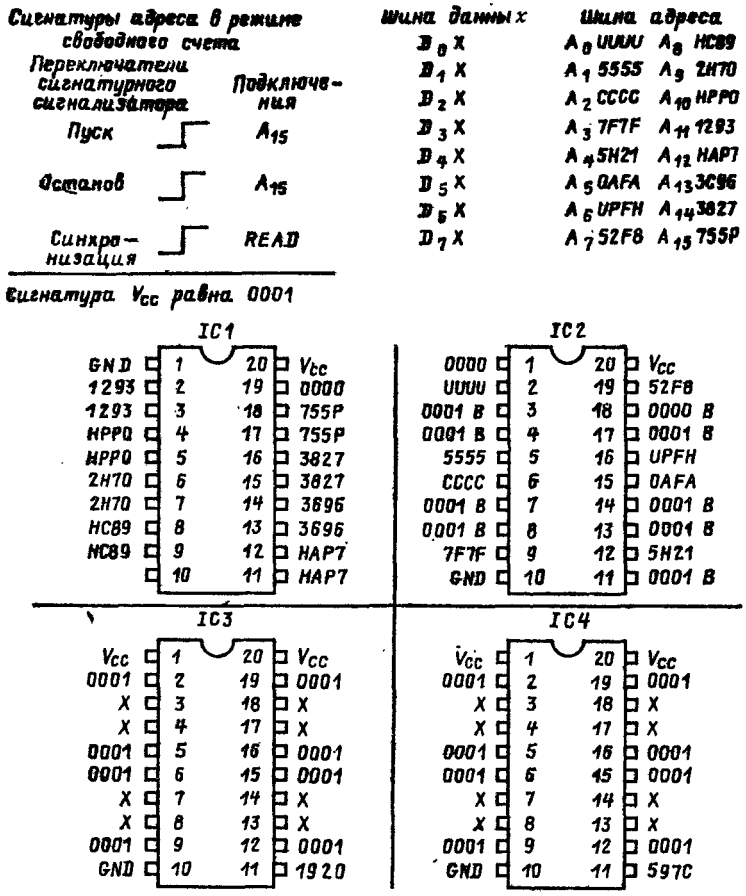



**Рис. 7.17.** Документирование сигнатур

ция отражена у вывода *1* ИС2. Здесь сигнатура равна 0000, но светодиод в зонде пробника не вспыхивает. Вывод *1* в данном тесте всегда имеет уровень логического 0, который дает такую же сигнатуру, как и земля; указание 0000 на диаграмме подчеркивает, что вывод не закорочен на землю. Если вывод закорочен на землю, следует указывать GND.

Поиск неисправности с применением сигнатурного анализа заключается в определении возможной области с неисправностью и проверке сигнатур до обнаружения

неправильной сигнатуры. Пользуясь принципиальной схемой и таблицами сигнатур, неисправность прослеживают «назад» до получения правильной сигнатуры. Неисправность существует в той части схемы, которая находится между последней неправильной и первой правильной сигнатурами.

### 7.9. Тестирование ПЗУ в режиме свободного счета

Хотя главное назначение свободного счета заключается в проверке системного ядра, он применим и для контроля ПЗУ. В режиме свободного счета на шине адреса периодически формируются все двоичные наборы. Подаваемая в ЦП холостая команда заставляет его выполнять операции считывания по каждому адресу. ПЗУ содержит только фиксированные команды, которые в режиме свободного счета последовательно выдаются на шину данных. Используя линию разрешения кристалла ПЗУ для сигналов пуска и останова, а управляющую линию $\overline{\text{READ}}$ для сигнала синхронизации анализатора, можно проверить содержимое любого системного ПЗУ.

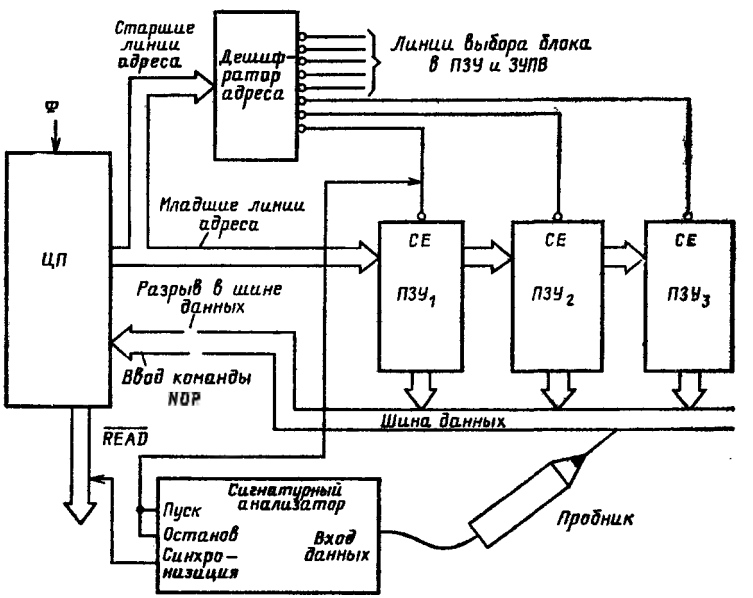



**Рис. 7.18.** Тестирование ПЗУ в режиме свободного счета

Анализатор регистрирует только данные, относящиеся к проверяемому ПЗУ, хотя процессор сканирует все адресное пространство. На рис. 7.18 показано, каким образом в режиме свободного счета проверяется одно из системных ПЗУ с применением сигнатурного анализатора.

Аналогичный тест для микросхем ЗУПВ применять нельзя, так как их содержимое не фиксировано, и для проверки работы ЗУПВ разработаны другие тесты. Поскольку ЦП выполняет только операции считывания из памяти, невозможно проверить каналы ВВ, особенно в том случае, если ВВ отображен на адресное пространство ВВ. Входные порты в системе с ВВ, отображенным на адресное пространство памяти, в режиме свободного счета проверить можно, так как ЦП считает обращения к ним операциями считывания из памяти. При этом необходимо управлять входами в порты, для чего обычно применяется тестовый прибор с возможностью задания известных двоичных наборов. Для проверки выходных портов потребуется операция записи, которой в режиме свободного счета нет.

### 7.10. Тест-циклы сигнатурного анализа

Для проверки тех частей системы, которые недоступны режиму свободного счета, необходимо написать и выполнить специальные программы. Каждая из них предназначена для проверки одной части системы, например входного порта, и обычно состоит всего из нескольких строк ассемблерного кода. Такие тест-программы сигнатурного анализа обычно помещаются в ПЗУ, которое находится в системе, но при нормальной работе не используется. Тест-ПЗУ приводится в действие либо переключением линии $\overline{CE}$ от первого системного ПЗУ с последующим сбросом системы, либо вводом в процессор команды RESTART и размещением тест-ПЗУ по адресу рестарта. Набор тест-программ обычно организован как цикл, который периодически выполняется при включении тест-ПЗУ в работу. В начале полного тест-цикла предусматривается формирование импульса на линии сигнала пуска сигнатурного анализатора, а в конце тест-цикла формируется импульс на линии сигнала останова анализатора. Часто для обоих сигналов пуска и останова используется одна и та же линия, что устраняет необходимость формирования отдельного сигнала останова.

В микропроцессорах типа 8080/Z80 в качестве сигнала пуска-останова обычно применяется старшая линия $A_{15}$ шины адреса, а само формирование сигнала осуществляется фиктивной командой ввода или вывода. В этих микропроцессорах адреса портов ВВ формируются только в младшей половине шины адреса, но адрес дублируется и на старшей половине шины. Поэтому считывание или запись в порт ВВ с адресом $80_{16}$ вызовет появление импульса на линии $A_{15}$.

```
TESTLOOP:   DI          ; запретить прерывания
            IN   80H    ; выдать импульс при
                        ; считывании
            OUT  80H    ; выдать импульс при
                        ; записи
```

Приведенную простую последовательность команд (в мнемониках микропроцессора 8080) можно использовать для инициирования набора тест-программ сигнатурного анализа. Команды IN и OUT вызывают появление импульса на линии $A_{15}$ сначала в операции считывания, а затем в операции записи. В зависимости от выполняемого теста вход синхронизации в сигнатурный анализатор подается с линий $\overline{\text{READ}}$ или $\overline{\text{WRITE}}$ шины управления.

Простой тест выходного порта реализуется следующим образом:

```
OUTEST:   XRA           ; сбросить регистр A
          STC     A     ; установить в 1 бит пе-
                        ; реноса
LOOP:     RAL           ; сдвинуть единичный
                        ; бит влево
          OUT     (04)  ; выдать двоичный на-
                        ; бор в выходной порт
          JNC     LOOP  ; повторить, если тест не
                        ; закончен
```

Принцип теста заключается в сдвиге состояния логической «1» по всем восьми линиям ВВ выходного порта. Таким образом, до перехода к следующему тесту будет произведено 8 операций записи в выходной порт. При использовании линии $A_{15}$ для сигналов пуска и останова, а линии $\overline{\text{WRITE}}$ для входа синхронизации берутся сигнатуры с каждой выходной линии и сравниваются с задокументированными значениями. Аналогичный тест можно написать для проверки каждой входной линии входного

порта, но вначале необходимо задать их известные состояния. Обычно для этого к входным линиям подключается тест-прибор, позволяющий устанавливать состояния входных линий.

### 7.11. Встроенные средства для сигнатурного анализа

В идеальном случае тестирование вычислительной системы с помощью сигнатурного анализа необходимо предусматривать еще на этапе проектирования. При разработке системы следует встроить средства реализации свободного счета и выбора тест-циклов из ПЗУ. Когда управляющие сигналы для анализатора должны формироваться из комбинации нескольких внутренних сигналов, следует ввести в систему необходимые схемы. После окончания разработки можно получить и задокументировать полные таблицы сигнатур для различных тестов.

Многие системы разработаны без всяких встроенных средств тестирования, но их можно приспособить для сигнатурного анализа. Прежде всего необходимо ввести схемы, реализующие режим свободного счета системного ядра, чтобы проконтролировать важнейшие компоненты компьютера. К счастью, в большинстве систем микропроцессор находится в панельке, и его можно вынуть. Панелька микропроцессора имеет все сигналы и напряжения питания, которые требуются для реализации свободного счета. В простейшем варианте нужно вынуть микропроцессор и вставить в его панельку переходник, в котором встроены разрыв шины данных и подача в микропроцессор холостой команды. После этого с другой стороны переходника вставляется микропроцессор. Для конкретного микропроцессора в отделе контроля нужно разработать переходник для встраивания в любую систему, в которой применяется данный микропроцессор. При эксплуатации нескольких систем с различными микропроцессорами потребуется изготовить соответствующие переходники для каждого микропроцессора. Таким образом, все возможности режима свободного счета реализуются в любой системе независимо от того, проектировалась она с учетом сигнатурного анализа или без учета его.

Тесты свободного счета проверяют только часть си-

стемы, а для стимулирования других ее частей необходимо написать и разместить в ПЗУ тест-программы. Обычный подход заключается в том, чтобы заменить первое системное ПЗУ на тест-ПЗУ. После сигнала системного сброса микропроцессор производит обращение к этому ПЗУ, и тест-программы проверяют всю систему.

Тест-программы сигнатурного анализа выполняются как последовательность коротких фрагментов, следующих друг за другом. По окончании последовательности процессор переходит к началу, образуя полный тест-цикл. Для локализации отказов в системе при проверке отдельных компонентов обычно требуются свои сигналы пуска и останова. Большинство этих сигналов уже имеется в системе; например, каждое ПЗУ можно проверить, подключив к управляющим линиям пуска и останова конкретный сигнал выбора кристалла от дешифратора адреса. ЗУПВ обычно проверяется путем выполнения первоначальной последовательности операции записи, в результате чего во всех ячейках ЗУПВ будет храниться известный двоичный набор. Операции записи контролируют линии шины адреса, подключенные к ЗУПВ, и линии шины управления, разрешающие работу микросхемы. Убедившись в функционировании этих линий, можно выполнить последовательность операций записи, которые контролируют линии подключения шины данных и проверяемого ЗУПВ. Любые сигналы, используемые в качестве сигналов пуска и останова сигнатурного анализатора, должны быть проверены предыдущими фрагментами программы. В идеальном случае все тесты должны опираться на результаты предыдущих тестов, контролирующих систему по частям. Режим свободного счета проверяет системное ядро, а работоспособное ядро обеспечивает подлежащие проверке сигналы. В большинстве случаев успешный контроль свободного счета свидетельствует о правильном функционировании самого микропроцессора.

### 7.12. Ограничения сигнатурного анализа

Сигнатурный анализатор регистрирует только те события, которые синхронны с сигналом синхронизации, используемым для получения сигнатур от узлов системы. Микропроцессор является синхронным конечным автоматом, и при сигнатурном анализе проверяется большин-

ство его частей. Однако некоторые события происходят асинхронно с системной синхронизацией, и их нельзя проконтролировать с помощью сигнатурного анализа. Последовательная линия связи в стандарте RS232C обычно работает асинхронно с системной синхронизацией, что препятствует применению сигнатурного анализа. В этом случае на вход синхронизации анализатора можно подать сигнал синхронизации универсального асинхронного приемопередатчика и получить сигнатуры последовательной линии связи. Однако компьютер необходимо запрограммировать на выдачу в линию периодического сигнала, чтобы в тесте участвовала известная периодическая двоичная последовательность.

Компьютер с шинной структурой можно считать замкнутой системой с обратной связью, в которой по шине адреса осуществляется запрос данных, а реакция вводится с шины данных. Проблема в такой структуре заключается в том, что отказ распространяется по всей петле, вызывая получение кажущихся плохих сигнатур в тех местах, где отказ отсутствует. Сигнатурный анализ не показывает, в каком месте временного окна впервые возникает отказ, что можно было бы использовать для локализации неисправного компонента. Плохая сигнатура по истечении временно́го окна не показывает «историю» прослеживания отказа. Поэтому, когда много микросхем осуществляют вывод на общую шину и появляется неправильная сигнатура, сигнатурный анализ не может указать, в какой именно микросхеме возник отказ.

Усовершенствованной формой сигнатурного анализа явился трассовый анализ, который локализует отказ в окне пуска-останова. Сигнатурный трассовый анализатор программируется (обычно вручную) на последовательность промежуточных сигнатур и их моменты появления во временно́м окне. При работе прибора первое несовпадение измеренных данных с хранимой информацией заставляет сигнатурный трассовый анализатор остановиться и индицировать момент появления ошибки. После этого отказ можно подозревать в той микросхеме, которая работала в момент его появления. Правильные данные для сигнатурного трассового анализа, как и сигнатуры, необходимо получить от заведомо исправной системы и задокументировать их. При этом сигнатуры разделяются на короткие секции с соответствующим увеличением объема документируемой информации. Однако

дополнительная документация потребуется только для тех частей системы, которые невозможно адекватно проверить, пользуясь обычным сигнатурным анализом.

### 7.13. Сигнатурный анализ как общий способ тестирования

Сигнатурный анализ представляет собой простой способ тестирования, заключающийся в проверке отдельных узлов схемы и сравнении их сигнатур с документированными значениями. Умелое его применение позволяет быстро идентифицировать неисправность в системе, а затем локализовать неисправный компонент.

Для сигнатурного анализа очень важна хорошая документация. Любые модификации в системе влекут за собой необходимость повторного получения всех сигнатур в системе, поэтому при ее модификациях и усовершенствованиях необходимо составлять скорректированные таблицы сигнатур. Важность поддержания своевременно обновляемой документации невозможно переоценить. Часто системы модифицируются, а изменения в принципиальные схемы не вносятся. Это обстоятельство само по себе создает проблемы тестирования, но, если не корректируются и таблицы сигнатур, сам способ оказывается неработоспособным и бесполезным, так как полученные сигнатуры не имеют смысла без соответствующих правильных документированных значений.

## 8. Эмуляция

Эмуляция — это процесс, в котором одна система используется для копирования действий другой системы. Имеется несколько уровней эмуляции разрабатываемой системы. В компьютере можно создать рабочую среду, которая предназначается для того, чтобы моделировать действия, ожидаемые в другой, независимой системе. Система ведет себя так, как будто она является другой системой, и поведение моделируемой системы изучается по действиям моделирующей системы. По существу, создается нереальная среда, которая, в общем, не ведет себя точно так, как система, которую она пытается моделировать. Обычно моделирующая система работает медлен-

нее моделируемой системы, так как в модели воспроизводятся действия аппаратных средств на основе использования программных аналогов. Моделирование дает только логическую модель разрабатываемой системы, и в ней отсутствуют рабочее быстродействие, а также реальные входы и выходы. Моделирование следует считать низшим уровнем эмуляции.

Другая крайность: можно построить конечную систему и воспользоваться диагностическими средствами, например логическим анализатором, для получения информации от системы для ее проверки и доказательства целостности. На этапе разработки прототипа, вероятно, имеется много возможных источников ошибок из-за отказов в аппаратных средствах и ошибок в программах. Система подвержена воздействию множества отказов, каждый из которых в отдельности является простым, но вместе они приводят к сложной ситуации, которую трудно диагностировать.

На практике предпочли использовать некоторую систему, которая позволила бы находиться где-то посередине между рассмотренными крайностями. Моделирование удобно для разработки программного обеспечения, но разработчики нуждаются также и в отладке аппаратных особенностей разрабатываемой системы. Например, может потребоваться использовать эмулирующую систему в качестве источника синхронизации разрабатываемой системы, но быть уверенным, что создаваемая система может работать и с собственной синхронизацией. Эмулирующую систему как диагностическое средство можно использовать для контроля работы целевой системы. Далее, удобно отлаживать программы целевой системы в среде эмулирующей системы и переносить их целыми блоками. По мере разработки каждой части программы ее можно проверить в рабочей среде и закончить загрузкой и контролем полной программы.

Возможность управлять действиями целевой машины из рабочей среды другой машины называется *внутрисхемной эмуляцией.* Этот термин введен фирмой Intel, которая впервые использовала внутрисхемную эмуляцию в качестве рабочего средства. Система, оснащенная этим средством, называется *системой проектирования.*

Системы проектирования обладают широким набором возможностей для разработки прототипа, а также диагностическими средствами для поиска неисправностей.

Приборы, в которых реализованы некоторые возможности системы проектирования для диагностики аппаратных отказов, называются автономными эмуляторами.

### 8.1. Системы проектирования

Система проектирования представляет собой микрокомпьютер, оснащенный средствами разработки аппаратных и программных средств. Для разработки программных средств предусмотрена совокупность программ, называемая операционной системой (ОС). Операционная система хранится на гибком (жестком) диске, и ее объем столь велик, что отдельные системные программы приходится загружать в основную память компьютера по мере их использования. Операционные системы, хранящиеся на диске, называются дисковыми операционными системами (ДОС). Типичная ДОС содержит следующие программы (этот список не окончательный): редактор, менеджер файлов, редактор связей/загрузчик, драйверы устройств ввода-вывода, ассемблер, компилятор, менеджер памяти, отладчик, анализатор реального времени, программатор ППЗУ.

В конкретной системе некоторые из этих программ называются по-другому, а некоторые могут быть объединены. Например, программа анализатора реального времени может быть частью программы отладчика.

В системе проектирования кроме стандартных аппаратных средств микрокомпьютера широкого назначения имеются такие дополнительные средства, как устройство управления памятью, трассовый анализатор реального времени, внутрисхемный эмулятор и программатор ППЗУ.

Приведенные выше средства образуют законченную систему проектирования, стоимость которой много выше стоимости микрокомпьютера широкого назначения. Все эти средства необходимы только в том случае, когда законченную целевую систему необходимо разработать в кратчайшие сроки. Имеется также несколько более дорогих универсальных систем проектирования, которые обеспечивают возможность разработки систем на основе нескольких микропроцессоров. В то же время фирмы-производители микропроцессоров предлагают системы проектирования, рассчитанные на конкретные микропроцессоры.

Операционная система обеспечивает взаимодействие пользователя с аппаратными средствами физической системы проектирования. Пользователь настраивает машину с учетом своих требований, вызывая те программы операционной системы, которые распознают и распределяют каналы ВВ периферийным устройствам, а также разделяют пространство памяти между самой системой проектирования и целевой машиной. В идеальном случае операционная система должна быть «настроена к пользователю дружески» в том смысле, чтобы изучать и пользоваться ею было легко. Первые операционные системы были далеки от этого идеала и требовали такого же долгого изучения, как и разрабатываемая микропроцессорная система. В новых системах этот недостаток был устранен с помощью «программируемых клавиш»: пользователь выбирает функцию, отмечая на экране ЭЛТ функции, назначаемые нескольким клавишам, которые находятся ниже индицируемых на экране символов. Например, при включении питания немаркированные программируемые клавиши можно обозначить EDIT, LOAD, ASSIGN, DEBUG и др. Когда пользователю требуется программа-редактор, он просто нажимает клавишу, отмеченную EDIT, и в основную память загружается программа-редактор. При работе с программой-редактором программируемые клавиши можно переопределить, пользуясь экраном, на функции NEW FILE, OLD FILE и QUIT. Если теперь пользователь хочет вызвать ранее написанную программу, он просто нажимает клавишу, помеченную OLD FILE, после чего на экран выводится просьба об указании имени файла. Программируемые клавиши устраняют необходимость изучения вызывающих кодов программ операционной системы и обеспечивают удобный интерфейс между пользователем и системой.

#### 8.1.1. Редактор

С помощью *программы-редактора* пользователь вводит новые программы или отдельные сегменты и запоминает их на каком-либо внешнем носителе, например гибком диске. Кроме того, ранее запомненные программы можно вызывать с диска и модифицировать. Таким образом, редактор оказывается основным средством ввода программ, которые в конце концов должны будут выполняться на целевой машине.

Программы вводятся на языке высокого уровня, например БЕЙСИК, ПАСКАЛЬ, или на языке АССЕМБЛЕР. Сам редактор не различает тип вводимого кода и не имеет средств контроля ошибок. Трансляцию с языка высокого уровня или ассемблирование осуществляют другие программы операционной системы, в которых предусмотрен контроль ошибок.

Когда программа запоминается в виде файла, к имени файла часто добавляется расширение, идентифицирующее тип файла. Если, например, на диске запоминается БЕЙСИК-программа с нменем ANALYSER, в некоторых системах она хранится как ANALYSER, BAS. Аналогично ассемблерная программа с тем же именем хранится как файл ANALYSER, ASM. Расширения символических имен файлов контролируют при своей работе и другие программы операционной системы. Так, например, если делается попытка компилировать ассемблерную программу, то на экран выводится сообщение об ошибке.

При модификацни программы большинство редакторов оперируют либо со строками, либо с экранным текстом. Экранный редактор всегда возбуждает максимальное число строк. Если, например, дисплей имеет 24 строки, то на экран выводятся 24 строки программы. Предусмотрены клавиши «рулона», позволяющие перемещать возбуждаемую (индицируемую) часть программы вперед или назад. Благодаря им пользователь быстро находит ту секцию программы, которую он хочет изменить. Найдя нужный фрагмент, пользователь удаляет или добавляет код, а также изменяет в программе имеющиеся значения. В экранном редакторе для позиционирования курсора в любом месте экрана имеются клавиши управления курсором. Предположим, что программист хочет ввести новую строку кода между ранее введенными строками. При помещении курсора на второй строке и задании режима INSERT редактор создает пустую строку между двумя строками на экране. После этого программист может печатать новую строку, создавая новый программный сегмент. Две старые строки могут находиться в любом месте экрана, и для идентификации второй из них применяются клавиши управления курсором. Курсор можно перемещать также по горизонтали, позиционируя его на знакоместо ранее введенной команды. После этого символ можно удалить, изменить или сдвинуть вправо для ввода дополнительного символа.

Строковый редактор образует на экране две горизонтальные линии, и данные разрешается вводить или модифицировать только между ними. Если, например, между двумя командами ранее написанной программы нужно ввести новую команду, вторую из них следует поместить между двумя линиями на экране. Клавиша «рулона» позволяет перемещать программу вверх и вниз экрана, и нужную позицию легко поместить в окне редактирования. Вызов затем режима INSERT заставит программу от второй команды сдвинуться вниз на одну строку и образовать в окне пустую строку. После этого в образованное пустое окно можно ввести новую команду. Когда необходимо отредактировать символ в строке, она вначале размещается в окне, а затем клавишами управления курсором идентифицируется то место, где необходимо произвести изменения.

Выбор между экранным редактором и строковым редактором зависит от привычки пользователя, и во многих системах имеются оба вида редакторов. Пользователь выбирает тот редактор, с которым, по его мнению, удобнее работать.

#### 8.1.2. Менеджер файлов

*Менеджер файлов* управляет операциями запоминания и загрузки между основной памятью компьютера и имеющейся в системе внешней памятью. Он может либо быть явной частью операционной системы и должен вызываться для загрузки в компьютер, например, программой-редактором, либо неявно присутствовать в операционной системе и вызываться для работы другими приказами, например, при вызове редактора одной из программируемых клавиш. Во втором случае менеджер файлов «невидим» пользователю и приводится в действие программой-супервизором.

В тех системах, где внешняя память не имеет достаточного «интеллекта», менеджер файлов реализует и общие служебные функции. К ним относятся поддержание карты адресов доступных блоков основной памяти и генерирование адресов связи, когда при запоминании программы приходится связывать несколько секторов.

### 8.1.3. Редактор связей/загрузчик

Полная прикладная программа системы обычно разрабатывается по сегментам, и причинами этого могут быть или большая длина программы, или участие нескольких программистов. Каждый сегмент запоминается, например, на гибком диске как отдельный файл. Сегменты представлены на языке высокого уровня или, как это часто бывает, в специализированных системах на языке АССЕМБЛЕР. Для превращения исходной программы в машинный код сегменты необходимо обработать программой-ассемблером. *Редактор связей* позволяет программисту указать программе-ассемблеру, какие файлы подлежат ассемблированию и в каком порядке. Имея возможность определять порядок ассемблирования, программисты могут разрабатывать программные сегменты в удобной для них последовательности.

Обрабатываемые ассемблером программы называются *исходными программами,* а получающаяся программа на машинном языке называется *объектной программой;* она автоматически запоминается на гибком диске как объектный файл. Программа-загрузчик предназначена для загрузки исполняемого объектного файла в память, начиная с определенной ячейки. Загруженную прикладную программу в машинном коде можно выполнять, если поместить в программный счетчик ЦП начальный адрес прикладной программы.

### 8.1.4. Драйверы устройств ввода-вывода

К системе проектирования подключаются разнообразные периферийные устройства — от *видеодисплея* до строчного принтера. Внешняя память также считается периферийным устройством. Для всех подключенных к системе проектирования устройств требуются специальные программы, которые воспринимают данные от системы и преобразуют их в формат, принятый для конкретного устройства. Пользователь должен распределить каналы ВВ между периферийными устройствами (видеодисплей, клавиатура), а только затем может пересылать информацию в устройства ВВ или получать ее от устройств ВВ. Например, видеодисплей можно обозначить меткой CONO и распределить (поставить в соответствие) каналу 0. После этого любая запись в канал 0 будет отображена на экране. Аналогично клавиатуру

можно обозначить CONIN и распределить (поставить в соответствие) каналу 1. Любое считывание из канала 1 вызовет ввод в программу символа с клавиатуры.

Видеодисплей и клавиатура, как правило, имеют номера каналов ВВ по умолчанию, которые при необходимости пользователь может изменить. Можно подключить любые другие устройства и оформить их программы-драйверы как часть операционной системы. Для этого в начале работы выполняется программа генерации системы, где драйвер устройства «встраивается» в операционную систему.

Драйверы устройств ВВ обеспечивают гибкое управление вводом-выводом и позволяют моделировать ВВ разрабатываемой системы с помощью стандартных периферийных устройств.

#### 8.1.5. Ассемблеры

Низшим уровнем, на котором можно программировать большинство микропроцессоров, является машинный код. Числа машинного кода представляют собой двоичные коды, которые микропроцессор дешифрирует для выполнения команд. Обычно при программировании пользуются 16-ричными значениями, которыми удобнее оперировать человеку. Программирование в машинном коде связано со значительным числом ошибок и трудностями внесения поправок при использовании абсолютной адресации. Каждый набор команд машинного кода имеет соответствующий набор мнемокодов языка АССЕМБЛЕР с однозначным соответствием между конкретной командой машинного кода и ассемблерной мнемоникой. Программирование с привлечением ассемблерных мнемоник эффективнее программирования в машинных кодах, так как возникает меньше ошибок из-за использования содержательных меток. Написанную в ассемблерных мнемониках программу необходимо преобразовать в машинный код, прежде чем ее можно выполнить в вычислительной системе. Эта задача возлагается на программу-ассемблер. Листинг программы в ассемблерных мнемониках с помощью редактора вводится в систему проектирования, а затем запоминается как файл во внешней памяти. Большинство ассемблеров имеют 4 поля, в которые вводится информация. Эти поля часто называются полями МЕТКИ, КОП, ОПЕРАНДА и КОММЕНТАРИЯ. Поле МЕТ-

КА позволяет обращаться к адресам по имени, а не по абсолютному адресу. Программы можно писать, пользуясь вместо адресов метками, а действительные адреса формируются только на заключительной стадии ассемблирования. В поля КОП и ОПЕРАНД вводятся ассемблерные мнемоники команд, а поле КОММЕНТАРИЙ позволяет программисту делать пояснения к программе с целью сделать ее более понятной. Знакомясь с программой, программист может разобраться в ее функциях по тщательно продуманным комментариям. Приведем простой пример ассемблерного листинга:

```
МЕТКА    КОП     ОПЕРАНД    КОММЕНТАРИЙ
START:   LD  A,  06H        ; загрузить в регистр A
         OUT (08H)          ; первое выводимое число
LOOP:    INC A              ; произвести инкремент числа
         OUT (08H)          ; числа
         JP  NZ,  LOOP      ; зациклить до получения нуля
```

В программе нет указаний прямых адресов, и программист должен помнить только мнемонические коды команд, а не 16-ричные числа в случае программирования на машинном коде. Отметим, что в листинге после чисел ставится буква *H*, информирующая ассемблер о представлении чисел в 16-ричном формате. Большинство ассемблеров допускают системы счисления с основаниями 2, 10 и 16. Поле комментария (обычно начинающегося точкой с запятой) не участвует в операции ассемблирования и предназначено только для помощи программисту.

Для каждого микропроцессора имеется соответствующая программа-ассемблер, которая преобразует конкретные мнемоники данного процессора в уникальные значения его машинного кода.

Ни один компьютер не сможет преобразовать приведенную выше программу в машинный код без дополнительной информации о том, в какой области памяти должна находиться программа. Эта важнейшая информация называется *директивой ассемблера*, и она должна быть задана до начала ассемблирования. Директивой начального адреса ассемблируемой программы почти во

всех ассемблерах служит мнемоника ORG (Origin — начало). Директива ORG 1000H заставит ассемблер поместить код операции первой команды программы по 16-ричному адресу 1000. Когда в программе встречаются команды перехода, ассемблер встраивает в программу машинные коды адресов в соответствии с определенным директивой ORG начальным адресом.

Ассемблеру необходимо сообщить, в каком месте программы следует закончить процесс ассемблирования программы. Для этого в тексте программы после последней команды помещается директива END. Имеются также директивы резервирования памяти для результатов промежуточных вычислений и (или) таблиц. С помощью соответствующей директивы ассемблера любой переменной, к которой в исходной программе имеются обращения по символическому имени, должно быть присвоено численное значение до начала ассемблирования. Например, входной порт, через который считываются состояния переключателей, в программе можно обозначить SWIT. Чтобы ассемблер мог преобразовать программу в машинный код, SWIT необходимо определить как численный адрес порта. Обычно эту задачу выполняет директива приравнивания EQU. Если, например, порт, обозначенный SWIT, в действительности является входным портом 04, перед началом ассемблирования следует использовать директиву SWIT EQU 04.

Многие ассемблеры называются *макроассемблерами,* так как они позволяют определять макрокоманды и включать их в машинный код объективной программы. Макрокоманда аналогична определяемой пользователем функции на языке высокого уровня; она записывается (определяется) 1 раз, а затем вызывается по имени. Например, можно напнсать макрокоманду умножения двух 16-битных чисел и назвать ее MULT. В исходной программе ассемблерная директива MACRO MULT заставит ассемблер ввести макрокоманду в данной точке прикладной программы. Макрокоманда отличается от подпрограммы тем, что она записывается 1 раз, но вводится как последовательность кодов при каждом ее вызове.

Ассемблеры широко используются для получения программ на машинном коде в системах управления. Ассемблерные мнемоники имеют однозначное соответствие с машинными кодами, поэтому ассемблерная программа

столь же эффективна, как и программа на машинном коде. Однако у ассемблерной программы появляется меньше ошибок из-за того, что она более понятна человеку.

### 8.1.6. Компиляторы

При программировании на машинном коде или на языке АССЕМБЛЕР программист должен помнить массу деталей об используемых в программе переменных и способах их обработки. Любую операцию приходится строить как последовательность простых команд вплоть до получения окончательного результата. Когда разрабатывается сложная программа силами нескольких программистов, то они должны договориться о типах данных, размещении данных в памяти и способах передачи данных между программными сегментами. Оказалось, что при этом людям проще и удобнее программировать на языках высокого уровня, операторы которых напоминают языковые конструкции предложений английского языка. В условиях дефицита времени программу можно написать на языке высокого уровня, например БЕЙСИК или ПАСКАЛЬ, а затем преобразовать в машинный код с помощью программы, называемой *компилятором*. Процесс трансляции реализуется либо непосредственно, либо через промежуточный язык, который иногда называется псевдокодом. Один из вариантов заключается в том, чтобы преобразовать программу в ассемблерные мнемоники, которые затем ассемблируются в машинный код.

Процесс трансляции довольно неэффективен, так как большинство компиляторов каждый оператор языка высокого уровня превращает в фиксированную последовательность ассемблерных мнемоник. Кодирование оказывается общим в том смысле, что оно используется всякий раз, когда один и тот же оператор встречается в программе на исходном языке. В каждом конкретном случае можно получить более эффективную программу, пользуясь непосредственно ассемблерными мнемониками. Типы данных обычно фиксированы, а это, в общем, приводит к распределению для переменной большего объема памяти, чем требуется. В большинстве компиляторов не предусмотрен поиск в программе повторяющихся частей, которые можно оформить в виде подпрограмм.

Компилированные программы занимают больше памяти и дольше выполняются, чем эквивалентные про-

граммы на языке АССЕМБЛЕР или на машинном коде. Во многих случаях увеличение объема требуемой памяти и времени выполнения не вызывает серьезных проблем, так как оно более чем компенсируется относительной простотой разработки прикладных программ.

Часто критичными по времени оказываются только отдельные фрагменты прикладной программы; например, может потребоваться программа задержки с точно заданным временем. В таких ситуациях большинство компиляторов допускают введение ассемблерных фрагментов. Поэтому критичные по времени исполнения фрагменты прикладной программы разрешается писать в ассемблерных мнемониках и при необходимости включать их в программу на языке высокого уровня.

Нет правил, устанавливающих размер программы, когда оправдано применение языка высокого уровня. Многие программисты предпочитают пользоваться ассемблерными мнемониками для программ почти любого размера из-за простоты доступа к данным, ее размещению в памяти и простоты манипуляций битами в слове. Программирование на языке АССЕМБЛЕР не составляет особых проблем при разработке программ объемом до 8К, но при его превышении следует внимательно проанализировать возможность применения языка высокого уровня.

#### 8.1.7. Менеджер памяти

Термин *менеджер памяти* относится к нескольким системным программам, одна из которых при необходимости динамически перераспределяет адресуемое пространство памяти. Менеджер памяти в системе проектирования предназначен для разделения системной памяти между самой системой проектирования и целевой машиной в процессах моделирования и эмуляции. В системе проектирования обычно применяется 8-битный микропроцессор с 16-битной шиной адреса, которая может прямо адресовать любую из 65 536 (64К) ячеек памяти. Чтобы иметь возможность работать с очень большими программами, например с компилятором, система комплектуется памятью максимальной емкости, и поэтому любой выданный ЦП адрес имеет соответствующую ячейку в памяти. Когда к системе проектирования подключается целевая система, которая имеет свою память, то она вступает в

конфликт с ячейками памяти системы проектирования. Для разрешения конфликта предусматривается селективное запрещение работы блоков памяти в системе проектирования и разрешение работы целевой системы. Таким образом, общее пространство памяти разделяется между двумя системами.

Возможность селективного запрещения работы блоков памяти выполняет двоякую функцию. На этапе моделирования память целевой системы блокируется, и для ее эмуляции используется часть памяти системы проектирования. В результате моделирующие программы выполняются в системе проектирования, и по их реакциям контролируется поведение моделируемой целевой системы. Очевидно, этот подход имеет ограничения, так как реальные аппаратные средства целевой системы не используются, процессор в системе проектирования может отличаться от процессора целевой системы, и между ними возможны различия во временных характеристиках. Однако моделирование полезно как средство отладки программных средств до их выполнения в целевой системе.

После отладки прикладных программ в системе проектирования их можно поместить в ЭИПЗУ, которое вставляется в целевую систему, или загрузить в ЗУПВ целевой системы из памяти системы проектирования. Чтобы предотвратить конфликты по адресам в обеих системах, вызывается менеджер памяти, который блокирует в системе проектирования адресное пространство, используемое целевой системой. Системе проектирования все же разрешается обращение к запрещенным ячейкам памяти, но она оперирует только ячейками, принадлежащими целевой системе. В этом случае целевая система выполняет находящиеся в ней программы со своим рабочим быстродействием, а система проектирования получает информацию о целевой системе путем опроса ее памяти. Такой режим работы может вызвать возможные конфликтные ситуации, так как при этом два устройства обращаются к общей памяти.

Часто системе проектирования разрешается обращаться к общей памяти только после появления определенных событий, например после контрольного останова в программе. Целевая система при этом переводится в режим останова, и система проектирования обращается к памяти для индикации ее состояния и состояния ЦП после контрольного останова.

Во многих системах проектирования основная системная память и память эмуляции полностью объединены, так что система проектирования может выполнять и свои программы, и операции эмуляции. В большинстве систем проектирования данного типа имеются два процессора, один из которых реализует такие функции, как выполнение программы менеджера памяти, а второй специализируется на моделировании и эмуляции.

#### 8.1.8. Отладчик

Системная программа, которая предназначена для локализации и исправления ошибок в программах, называется *отладчиком*. Отладчик в системах проектирования состоит из нескольких частей; на низшем уровне он выполняет функции простой программы-монитора, а на высшем уровне отладчик прослеживает и индицирует операции целевой системы вплоть до задания контрольных остановов.

Типичным для мониторного уровня является приказ индикации D, который осуществляет вывод на экран содержимого указанных ячеек памяти. Так, приказ индикации ячеек памяти 0070...008F имеет вид D 0070 008F.

С помощью отладчика можно изменить содержимое ячеек памяти. Под управлением отладчика выполняется модификация программы.

Эти операции необходимы в том случае, когда после ассемблирования и выполнения программы в ней обнаруживаются ошибки, которые можно исправить путем изменения содержимого определенных ячеек памяти; после исправлений осуществляется повторное выполнение программы. Особенно удобно пользоваться средствами отладчика на этапах разработки и моделирования.

В процессе эмуляции система проектирования применяется для прослеживания работы целевой системы без непосредственного взаимодействия с ней. На этом этапе используются специальные приказы, отличающиеся от тех, которые использовались при отладке программ. Отладчик позволяет установить точку контрольного останова прикладной программы, и все действия целевой системы при выполнении программы до контрольного остаиова будут зарегистрированы в памяти. Из-за ограниченной емкости памяти системы проектирования удается запомнить только ограниченное число событий.

Кроме задания контрольных остановов и прослеживания хода исполнения программы отладчик позволяет также выполнять программу в целевой системе в пошаговом режиме (по командам), определять режим эмуляции и программировать ЭИПЗУ.

#### 8.1.9. Анализатор реального времени

Во многих системах проектирования имеется *логический анализатор*, который регистрирует информацию от целевой системы так же, как и автономный анализатор. Обычно в анализаторе есть своя память, но программируется он с клавиатуры системы проектирования и индицирует результаты измерений на экране видеодисплея. Так как анализатор является только частью всей системы, его возможности значительно ниже по сравнению с автономными логическими анализаторами.

#### 8.1.10. Программатор ЭИПЗУ

В ходе разработки системы коды прикладной программы записываются в микросхемы ПЗУ, которые вставляются в целевую систему, и только после этого прикладная программа выполняется в ее окончательной рабочей среде. При массовом выпуске изделия применяются ПЗУ, программируемые маскированием, но на этапе разработки обычно пользуются микросхемами ЭИПЗУ, которые позволяют вносить изменения в программы. В системе проектирования имеются средства программирования ЭИПЗУ, которые могут быть представлены или отдельной программой в операционной системе, или специальной программой в составе отладчика.

#### 8.1.11. Внутрисхемные эмуляторы

Чтобы осуществить эмуляцию целевой системы в системе проектирования, эти системы необходимо объединить с помощью *переходника эмулятора*. Микропроцессор вынимается из целевой системы и помещается в переходник, который затем вставляется в свободную панельку микропроцессора. Все сигналы микропроцессора через буферы подаются в систему проектирования, которая может также управлять микропроцессором с помощью задания сигналов на его шине управления. Информация воспринимается с шин целевой системы и запо-

минается в памяти эмуляции системы проектирования.

Одной из главных задач любого эмулятора является согласование быстродействия целевой машины со средой эмуляции в системе проектирования. Переходник эмулятора удлиняет сигнальные тракты, что может привести к временным перекосам сигналов в обеих системах. По мере появления все более быстродействующих микропроцессоров необходимо разрабатывать соответствующие по быстродействию системы эмуляции. Это относится не только к схемам переходника и буферных каскадов, но и к памяти системы проектирования, которая предназначена для хранения информации от целевой системы. Для любого микропроцессора приходится разрабатывать соответствующий эмулятор, который может регистрировать информацию и управлять ею в реальном времени. Реальное время в данном контексте определяется целевой системой, а не относительно низким быстродействием системы проектирования.

Информация передается между двумя системами, причем целевая система работает в псевдонормальных условиях: ее микропроцессор находится в переходнике эмулятора, но системе кажется, что микропроцессор работает в собственной панельке. Такую среду эмуляции фирма Intel называет *внутрисхемной эмуляцией.* При повышении частоты синхронизации и ускорении передач информации разработка систем внутрисхемной эмуляции становится сложной задачей.

## 8.2. Тестирование систем с применением системы проектирования

Основные цели любой системы проектирования — разработка и отладка аппаратных и программных средств на всех этапах проектирования и создание прототипного образца изделия. После разработки ту же самую систему проектирования можно применить и для поиска неисправностей в целевой системе. В системе проектирования предусматриваются программы, с помощью которых ведется поиск неисправностей в целевой системе с использованием переходника эмуляции. В процессе поиска неисправностей участвуют программы отладчика и анализатора реального времени. К ним можно добавить программы *сигнатурного анализа,* выполняемые из системы проектирования. Такие расширения требуют

модификации переходника эмулятора для создания условий свободного счета и разработки тест-циклов сигнатурного анализа. Последние хранятся в системе проектирования и используются для проверки различных компонентов целевой системы. Тест-циклы обычно предусматривают формирование в целевой системе сигналов пуска, останова и синхронизации, а для регистрации и индикации сигнатур требуется отдельный сигнатурный анализатор.

Персоналу, связанному с тестированием систем, система проектирования обычно недоступна, а при ее наличии маловероятно, что она будет именно той системой, которая применялась при разработке изделия. Следовательно, тест-программы для контроля неисправного изделия получить не так легко. Это ограничение частично снимается благодаря появлению дешевых (по сравнению с системами проектирования) автономных эмуляторов.

### 8.3. Автономные эмуляторы

Если удалить микропроцессор из системы и вставить его в эмулятор, можно управлять системой и испытывать ее через эмулятор. Этот принцип лежит в основе систем эмуляции. Управляющий элемент — микропроцессор — теперь находится в отдельной среде, которую персонал может программировать для испытаний целевой системы и фиксации сообщений о всех обнаруженных в ней отказах.

В эмуляторе можно разорвать шину данных и ввести в микропроцессор разновидность холостой команды, которая переводит его в режим свободного счета. Следовательно, с помощью эмулятора проверяются ядро системы и линии шины данных. Открытый конец шины данных также доступен эмулятору, что позволяет выполнить тесты ПЗУ в режиме свободного счета. Усовершенствованный эмулятор не требуется информировать о карте памяти целевой системы, так как он может определить области ПЗУ и ЗУПВ, зная процессор и выполнив тесты шахматной доски.

В эмуляторе необходимо предусмотреть несколько тестов ЗУПВ. Функциональный контроль ячеек ЗУПВ осуществляется стандартными тестами шахматных наборов: в ячейки ЗУПВ записываются и считываются

двоичные наборы 01010101 и 10101010. Во многих эмуляторах имеются тесты ЗУПВ с набором 00000000 и 11111111.

Некоторые отказы ЗУПВ возникают из-за коротких замыканий линий шины адреса внутри микросхемы или вне ее. Поэтому во многих эмуляторах имеется тест для проверки подобных отказов. Первоначально все ячейки ЗУПВ сбрасываются, а затем в одну ячейку записывается определенный двоичный код, например $FF_{16}$. После этого производится считывание из всего ЗУПВ с проверкой того, не содержится ли записанный тест-код в какой-то другой ячейке. Если тест-код обнаруживается в другой ячейке, ее адрес объединяется операцией исключающего ИЛИ с адресом той ячейки, в которую первоначально был загружен тест-код, в результате чего идентифицируются подозрительные линии шины адреса.

Предположим, что содержимое ЗУПВ сброшено и в ячейку с адресом $0800_{16}$ записан код $FF_{16}$. Пусть программа считывания всего ЗУПВ обнаруживает тест-код по адресу $0900_{16}$. Тогда объединение операций исключающего ИЛИ адресов $0800_{16}$ и $0900_{16}$ показывает короткое замыкание между линией $A_8$ и землей:

| | $A_{15}$ | | | $A_0$ |
|---|---|---|---|---|
| Адрес 0800 | 0000 | 1000 | 0000 | 0000 |
| Адрес 0900 | 0000 | 1001 | 0000 | 0000 |
| Исключающее ИЛИ | 0000 | 0001 | 0000 | 0000 |

↑ Этот бит показывает неисправность на линии $A_8$ шины адреса.

Предположим далее, что закорочены линии $A_8$ и $A_9$. Тогда запись тест-кода в ячейку $0900_{16}$ приведет к считыванию тест-кода из ячейки $0B00_{16}$. Операция исключающего ИЛИ покажет наличие неисправности на линии $A_9$:

| | $A_{15}$ | | | $A_0$ |
|---|---|---|---|---|
| Адрес 0900 | 0000 | 1001 | 0000 | 0000 |
| Адрес 0B00 | 0000 | 1011 | 0000 | 0000 |
| Исключающее ИЛИ | 0000 | 0010 | 0000 | 0000 |

↑ Этот бит показывает неисправность на линии $A_9$ шины адреса.

Отказ можно локализовать на самой шине адреса или в конкретной микросхеме, проверив линии шины адреса

в режиме свободного счета. Если отказ проявляется только во время теста ЗУПВ, он скрыт в микросхеме ЗУПВ, которую необходимо заменить.

Эмуляторы помимо запрограммированных в них фиксированных тестов допускают выполнение тест-программ, написанных пользователем. Их можно хранить в съемном ЭИПЗУ, но можно разработать и на самом эмуляторе и передать на кассету или гибкий диск. Сами тест-программы могут быть рассчитаны на сигнатурный анализ или на проверку отдельных компонетов системы. Таким образом, эмуляторы превращаются в универсальное средство, реализующее режим свободного счета, программы сигнатурного анализа и общие диагностические программы для проверки системы. Обычно они достаточно портативны и более привлекательны для обслуживающего персонала, чем системы проектирования.

## 9. Программы самоконтроля и диагностики

По мере усложнения оборудования первостепенное значение приобретает необходимость встроенного самоконтроля. «Идеальная» система при включении питания осуществляет самоконтроль и не запускается при обнаружении отказа. Она должна также указать подозрительную секцию системы, чтобы ее можно было заменить и минимизировать время простоя. В большой системе самоконтроль необходимо проводить только до уровня подсистем с проверкой общих функций по принципу «проходит/не проходит». Для каждой функции необходимо предусмотреть какой-нибудь простой индикатор, например светодиод, показывающий ее состояние, чтобы первое действие заключалось в проверке светодиодов и обнаружении подозрительной подсистемы.

Хотя этим способом можно проверить множество функций микрокомпьютера, подсистемы ВВ, особенно те, которые постоянно подключены к другому оборудованию, проверить без испытаний линий ВВ почти невозможно. Процедура тестирования может оказаться опасной для подключенного к компьютеру оборудования. Такие системы можно проверить только при наличии дополнительных схем, которые при проведении теста не

позволяют тест-сигналам оказывать нежелательные воздействия на внешнее оборудование. Очевидно, возникнут проблемы контроля дополнительных схем, в которых также может возникнуть отказ.

К сожалению, большинство разработчиков концентрируют свои усилия на главных функциях системы и часто рассматривают тестирование как дополнительную обузу. Поэтому тест-программы и дополнительные схемы, необходимые для производства тестов, считаются излишествами. Требующиеся схемы могут повлечь значительные переделки и не вводятся в систему. В большинстве современных микропроцессорных систем либо имеются минимальные средства тестирования, либо их совсем нет; создавшееся положение должны поправлять специалисты по тестированию. Тест-переходники для реализации свободного счета и диагностические программы должен разрабатывать персонал, которому требуется значительное время на изучение системы до такой степени, чтобы суметь написать тест-программы. Эти функции должны быть встроены в систему на этапе проектирования разработчиками, которые досконально знают систему.

Проблемы приспосабливания средств тестирования приводят многих пользователей к требованию встраивания этих средств в систему с самого начала. Следовательно, разработчики должны предусматривать в системах такие средства тестирования, которые удовлетворяют потребности заказчиков в удобной профилактике и сокращении времени простоя. Однако эти утопические пожелания не относятся к очень большому числу уже выпущенных изделий с микропроцессорами. Для таких изделий необходимо разработать средства самоконтроля и написать диагностические тест-программы.

### 9.1. Программы самоконтроля

К программам самоконтроля относятся те программы, которые инициируются при включении электропитания системы и контролируют ее в естественной рабочей среде. В них не включаются тесты функционирования ВВ, где сами тест-наборы могут создать потенциально опасные ситуации. В программах самоконтроля при включении питания обычно предполагается работоспо-

собность системного ядра, и в противном случае они не действуют. В основном программы самоконтроля проверяют микросхемы памяти, охватывая ПЗУ и ЗУПВ. Если тесты проходят, считаются работоспособными схемы дешифрирования адреса и шина данных. ПЗУ, содержащее программы самоконтроля, должно функционировать правильно (или, по крайней мере, та его секция, в которой находятся тест-программы). Хотя на первый взгляд перечисленные условия кажутся очень жесткими, все же в случае отказа при включении питания подозреваемая область сужается до нескольких микросхем. Подозрительные микросхемы проверяются по отдельности в режиме свободного счета.

Все тест-программы должны быть короткими и простыми. Вместо попытки охватить сразу несколько микросхем следует написать короткую отдельную программу для каждой микросхемы. Основная причина следования этому простому правилу заключается в том, что в короткой программе больше гарантий отсутствия ошибок и на нее можно больше полагаться во время тестирования. Данного принципа следует придерживаться в любой тест-программе, будь это программа самоконтроля, диагностическая программа или программа сигнатурного анализа.

#### 9.1.1. Тестирование ПЗУ

Простейшим тестом микросхем ПЗУ является вычисление контрольной суммы, при котором суммируется содержимое всех ячеек ПЗУ, а возникающие переносы игнорируются. Последняя ячейка ПЗУ не участвует в сложении и предназначается для хранения результата сложения. В конце полученная сумма сравнивается с содержимым последней ячейки. Если они совпадают, предполагается, что ПЗУ полностью исправно; в случае расхождения имеется отказ, что приводит к включению соответствующего индикатора ошибки. В другом варианте этого способа тестирования значение контрольной суммы хранится в отдельной микросхеме ПЗУ, а не в последней ячейке проверяемого ПЗУ. В системе с большим числом микросхем ПЗУ второй вариант предпочтительнее, так как при этом не нужно предусматривать для каждой микросхемы команду перехода, с помощью которой «обходится» ячейка контрольной суммы.

Типичная тест-программа ПЗУ имеет следующий вид:

```
Начало теста ПЗУ
START:    LD   A, 00          ; сбросить аккумулятор и регистр B
          LD   B, A           ; гистр B
          LD   HL, ROMTOP     ; начальный и конечный адреса
          LD   DE, ROMBOT     ; ПЗУ в HL и DE
ROMLOOP:  DEC  HL
          LD   A, (HL)        ; считать содержимое ПЗУ
          ADD  B              ; прибавить сумму из регистра B
          LD   B, A           ; передать сумму в регистр B
          XOR  A, A           ; сбросить флажок переноса
          SBC  HL, DE         ; сравнить адреса
          JP   NZ, ROMLOOP    ; зациклить
          LD   HL, ROMTOP     ; адрес контрольной суммы в HL
          LD   A, B           ; полученная сумма в A
          CP   (HL)           ; сравнить итог с контрольной суммой
          JP   NZ, ROMFAULT   ; если различаются, перейти
                              ; к процедуре ошибки, иначе
                              ; продолжать тест
Конец теста ПЗУ
```

Программа написана в ассемблерных мнемониках микропроцессора Z80 и применима для любой микросхемы ПЗУ в системе. Для ассемблирования необходимы директивы приравнивания символических адресов ROMTOP и ROMBOT действительным значениям адресов. Если, например, проверяется ЭИПЗУ емкостью 2К байт типа 2716 и его адресное пространство в системе составляет $0000 \ldots 07FF_{16}$, перед началом ассемблирования необходимо ввести такие директивы:

```
ROMTOP    EQU    07FF
ROMBOT    EQU    0000
```

Программа начинается со сброса регистра A (аккумулятора) и регистра B. Регистр B предназначен для

хранения накапливаемой суммы по мере сложения содержимого ячеек ПЗУ. Регистровая пара HL адресует ячейку ПЗУ с максимальным адресом, которому присвоено имя ROMTOP. Регистровая пара DE зарезервирована для хранения минимального адреса ПЗУ, которому присвоено имя ROMBOT. Предполагается, что ROMTOP содержит значение контрольной суммы. Затем программа входит в цикл сложения, где в регистре B накапливается сумма содержимого всех ячеек ПЗУ, за исключением ROMTOP. Когда содержимое HL доходит до значения ROMBOT, цикл заканчивается, и в HL вновь загружается адрес ROMTOP. Значение из адресуемой HL ячейки сравнивается с суммой, накопленной в регистре B, и, если они различаются, программа переходит к процедуре индикации ошибки, названной ROMFAULT.

Аналогичную программу с соответствующими мнемониками можно написать для любого микропроцессора. Некоторые действия потребуют дополнительных команд, если прямых эквивалентов не существует. Адреса для ROMTOP и ROMBOT также зависят от микропроцессора. Микропроцессоры фирмы Intel и микропроцессор Z80 фирмы Zilog рассчитаны на размещение ПЗУ в начальной части карты памяти, так как при включении питания программный счетчик устанавливается в нуль. В микропроцессоре 6802 фирмы Motorola и в микропроцессоре 6502 фирмы MOS Technology ПЗУ размещаются в конечной части адресного пространства памяти.

Если контрольная сумма хранится в отдельном ПЗУ, программу необходимо модифицировать: в цикл сложения следует включить самую верхнюю ячейку и соответственно учесть адрес ячейки, в которой хранится контрольная сумма.

#### 9.1.2. Тестирование ЗУПВ

Основной тест ЗУПВ заключается в записи «тяжелого» двоичного кода ($55_{16}$ или $AA_{16}$) в каждую ячейку и последующего считывания. Если содержимое какой-либо ячейки отличается от записанного кода, считается, что возник отказ, и включается соответствующий индикатор ошибки. На практике вместо «тяжелых» кодов применяется код $00_{16}$. Предполагается, что содержимое ЗУПВ при включении питания произвольно, поэтому вероят-

ность установки ячеек в состояния $55_{16}$, $AA_{16}$ и $00_{16}$ одинакова. Применение кода $00_{16}$ устанавливает содержимое ЗУПВ в нуль до его использования системой.

Общая программа тестирования ЗУПВ имеет следующий вид:

*Начало теста ЗУПВ*

```
RAMTEST:    LD    A, C0          ; сбросить аккумулятор
            LD    HL, RAMBOT     ; начальный и конечный адреса
            LD    BC, RAMTOP     ; ЗУПВ в HL и BC
RAMLOOP:    LD    (HL), A        ; записать 00 в ячейку ЗУПВ
            LD    A, (HL)        ; считать из ячейки ЗУПВ
            CP    00             ; проверить результат на 00
            JP    NZ, RAMFAULT   ; если не 00, перейти к процедуре
                                 ; ошибки ЗУПВ
            INC   HL             ; перейти к следующей ячейке
            XOR   A, A           ; сбросить бит переноса
            SBC   HL, BC         ; сравнить адреса
            JP    NZ, RAMLOOP    ; зациклить
```

*Конец теста ЗУПВ*

Вначале программа, написанная в ассемблерных мнемониках микропроцессора Z80, сбрасывает регистр A в нуль. Именно это значение записывается в каждую ячейку ЗУПВ. Затем в регистровую пару HL загружается адрес нижней ячейки, названный RAMBOT, а в регистровую пару BC загружается адрес RAMTOP самой верхней ячейки ЗУПВ. Цикл, начинающийся с метки RAMLOOP, записывает нуль в ячейку, а затем считывает его. Если результат операции считывания не равен нулю, то в ЗУПВ возник отказ и программа переходит к процедуре ошибки RAMFAULT. Когда же из ячейки считывается нуль, производится инкремент регистровой пары HL и цикл записи-считывания повторяется. По окончании проверки всех ячеек пара HL адресует ячейку RAMTOP, и программа заканчивается.

Приведенную выше программу можно использовать

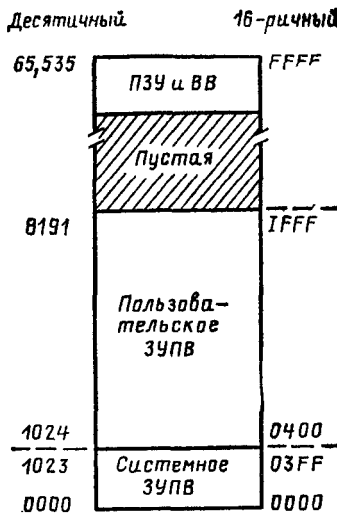



**Рис. 9.1.** Карта памяти системы

в микрокомпьютере широкого применения для нахождения верхней ячейки ЗУПВ. При установке пары HL для указания на нижнюю ячейку ЗУПВ и исключении из программы команды загрузки регистровой пары BC программа продолжает прохождение памяти до тех пор, пока она не встретит ячейку ЗУПВ, не содержащую нуль, или не найдет пространство памяти, которое в системе не занято. Предположим, например, что карта памяти системы имеет вид, представленный на рис. 9.1. Такая карта типична для небольшого микрокомпьютера, реализованного на основе микропроцессоров 6802 фирмы Motorola или 6502 фирмы MOS Technology. Здесь ПЗУ находится в верхней части пространства памяти вплоть до ячейки $FFFF_{16}$ (65 535). Нижняя часть памяти содержит ЗУПВ, которое зарезервировано для использования самой системой. В ячейках этой памяти находятся часто используемые константы и системный стек. Пользовательская память начинается с ячейки $0400_{16}$ и продолжается до $1FFF_{16}$. Таким образом, система содержит ЗУПВ 8К, из которых пользователю доступны только около 7К.

Программу, аналогичную программе тестирования ЗУПВ, можно применить для установки ЗУПВ в нуль и для определения его емкости. Следует определить RAMBOT как $0400_{16}$, и программа запишет и считает нуль из каждой ячейки пользовательского ЗУПВ до достижения ячейки с адресом $2000_{16}$. Эта ячейка находится за пределами адресов ЗУПВ. Считывание дает ненулевое значение, и программа полагает, что достигнута граница ЗУПВ. Первая ячейка, где тест записи-считывания нуля не проходит, имеет десятичный адрес

8192. Из этого значения нужно вычесть объем системного ЗУПВ в нижней части памяти и получить, таким образом, объем пользовательского ЗУПВ 8192—1023= =7169 байт. Многие микрокомпьютеры показывают это значение на экране видеодисплея для проверки емкости доступного системе ЗУПВ. Если в систему подключить дополнительное ЗУПВ емкостью 8К, верхний адрес будет $2FFF_{16}$ и тест-программа покажет значение 16 384— —1023=15 361 байт.

Данный способ позволяет локализовать и индицировать неисправную микросхему ЗУПВ. Предположим, что имеется отказ в ячейке $081E_{16}$ (2079). Считывание из этой ячейки ненулевого значения вызовет окончание тест-программы и индикацию на экране значения 2079—1023=1056. Очевидно, оно намного меньше обычного значения 7169 байт, характерного для системы с памятью 8К. Зная емкости применяемых в системе микросхем ЗУПВ, пользователь может определить неисправную микросхему ЗУПВ и исследовать ее.

Во многих небольших микрокомпьютерах имеется видео-ЗУПВ, каждая ячейка которого однозначно отображается в символьную позицию экрана видеодисплея. ЗУПВ видеосистемы можно проконтролировать с помощью тест-программы, но остальные части схемы дисплея процессор непосредственно проверить не может. Любое устройство отображения реализует только операции записи и не имеет явной обратной связи в вычислительную систему. Тест выполняется путем вывода на экран эталонного изображения и обращения к оператору проверить его правильность нажатием клавиши. В этом случае тестирование системы включает в себя участие оператора для тех компонентов, которые трудно проверить автоматически.

Тестирование ввода-вывода средствами самоконтроля при включении питания можно осуществить только частично из-за вероятности установки сигналов выходных линий в нежелательные и, возможно, опасные состояния, а также из-за отсутствия обратной связи от внешних линий в микрокомпьютер. Входные порты проверяются посредством подачи в них известных стимулирующих наборов, которые компьютер считывает и идентифицирует как правильные или неправильные. Стимулирующий набор либо подается от тест-прибора, который подключается на вход по запросу вычислительной

системы, либо выбирается под управлением компьютера во время последовательности самоконтроля. При определенных условиях можно проверить и некоторые выходные функции, не воздействуя на внешние схемы, если запретить работу оконечных схем через управляемые компьютером тристабильные буферы. Выходной двоичный набор следует подать во входной порт и проверить его правильность. Подробнее о тестировании ВВ см. гл. 10.

При автоматическом тестировании ВВ в системе обычно требуются либо дополнительные аппаратные средства, которые защищают внешние схемы, управляемые компьютером, либо средства подключения стимулирующих устройств по запросу компьютера. Второй вариант обычно реализуется вместе с диагностическими тест-программами, которые либо постоянно хранятся в системе, либо загружаются и инициируются обслуживающим персоналом.

### 9.2. Диагностические тесты

Когда система частично не работает, а исправность ядра проверена в режиме свободного счета, пользователь может вызвать диагностические тест-программы для проверки компонентов компьютера. Тесты обычно хранятся в ПЗУ, а само ПЗУ либо постоянно находится в системе, либо вставляется в специально предусмотренную панельку. Но тесты можно также загрузить из внешней памяти, например с кассеты или гибкого диска. Во многих случаях первый способ предпочтительнее, так как отказы в периферийных схемах компьютера могут препятствовать загрузке тест-программ.

Разомкнутая структура большинства функций ВВ предполагает участие в тестах пользователя, контролирующего правильность работы схем. Примером служат тест-программы сигнатурного анализа — пользователь должен подключить сигнатурный анализатор к компьютеру и сравнить получающиеся сигнатуры с сигнатурами из документации. При отсутствии таких программ и документации можно реализовать простые тесты, проверяющие различные компоненты системы. Большинство вычислительных систем подключено к внешним устройствам через какой-либо разъем, который можно отсое-

динить, а вместо устройства подключить тест-прибор. Последний выполняет функции стимулирующего устройства и обычно имеет переключатели для стимулирования входов и индикаторы для отображения выходных значений. Когда в системе имеются клавиатура и видеодисплей, они сами могут служить тест-прибором, выполняя функции разрешения работы входов и выходов системы, а также реализуя тестирование функций клавиатуры и дисплея. Например, диагностическая тест-программа может вывести на экран сообщение о необходимости последовательного нажатия всех клавиш на клавиатуре. Затем каждый символ выводится на экран, и компьютер предлагает пользователю проверить их правильность и затем нажать определенную клавишу. Конечно, при применении этого способа есть недостаток, заключающийся в том, что клавиша контроля может оказаться неисправной. При использовании тест-прибора компьютеру необходимы конкретный стимулирующий набор, который он может проверить как правильный или неправильный, а также какая-то обратная связь от оператора для подтверждения правильности посылаемых в выходные порты двоичных наборов.

В больших компьютерах применяется несколько уровней диагностических программ. На первом уровне система считается состоящей из функциональных подсистем и осуществляется контроль правильной работы каждой подсистемы. Подсистемами могут быть накопитель на гибком диске или модуль сбора данных. В случае отказа вызываются тесты нижнего уровня, которые более детально проверяют неисправную подсистему. Очевидно, что при отказе существует граница, до которой можно провести дальнейшее тестирование, и детальные тесты зависят от особенностей проверяемой подсистемы.

#### 9.2.1. Простое тестирование ввода-вывода

В большинстве вычислительных систем есть минимум один входной порт и один выходной порт, реализованные на простых ТТЛ-схемах или программируемых микросхемах ВВ. Их функционирование проверяется путем подсоединения переключателей ко входному порту и светодиодных индикаторов к выходному порту. Следующая программа показывает характер выполняемого теста:

*Начало простого теста ВВ*

```
/OTEST:     XOR   A                    ; сбросить регистр
                                       ; A
            OUT   (OUTPORT), A         ; выключить все
                                       ; светодиоды
I/OLOOP:    IN    A, (INPORT)          ; ввести состояния
                                       ; переключателей
            OUT   (OUTPORT), A         ; вывести на све-
                                       ; тодиоды
            CP    81H                  ; установлены
                                       ; младший и стар-
                                       ; ший биты?
            JP    NZ, I/OLOOP          ; если нет, ввести
                                       ; иовый входной
                                       ; набор
```

*Конец простого теста ВВ*

В программе предполагается, что тест-прибор подключен к входному порту INPORT, а набор светодиодов — к выходному порту OUTPORT. Вначале выключаются все светодиоды, а затем считываются состояния переключателей, которые выводятся на светодиоды. Каждый входной и соответствующий выходной бит можно проверить для обоих логических состояний путем коммутации переключателя в данном бите. Чтобы обеспечить окончание программы, она проверяет, что переключатели в младшем и старшем битах установлены одновременно. При удовлетворении этого условия осуществляется завершение программы. Данная программа может быть первой программой в ПЗУ диагностических тестов, которое устанавливается вместо первого системного ПЗУ и инициируется при сбросе системы.

Иногда проверку можно автоматизировать, если имеется возможность подсоединить выходной порт к входному и выполнить программу, которая проверяет каждый бит. Программа контролирует, что выданный двоичный набор совпадает с принятым, и индицирует отказ при рассогласовании. Некоторые микросхемы, например микросхема ВВ фирмы Zilog, уже имеют такое соединение обратной связи в каждом из своих программируемых портов ВВ. Следовательно, порт можно запрограммировать на вывод, сохранив работоспособным соответствующий входной порт. Данные можно записать в выходной порт и считать в компьютер по тому же адресу ВВ. Хотя этот способ проверяет линии ВВ компьютера, его можно реализовать только при условии защиты ли-

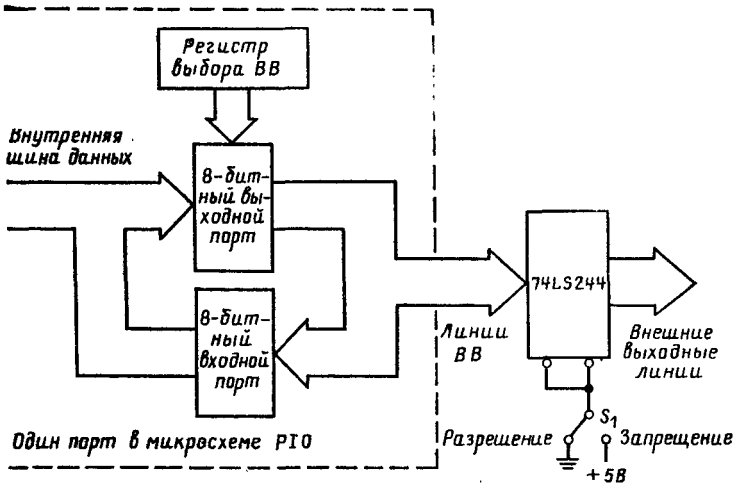



**Рис. 9.2.** Тестирование программируемой микросхемы ввода-вывода

ний, идущих в периферийное устройство. Для этого следует либо совсем отключить периферийное устройство, либо перевести выходные буферы в высокоимпедансное состояние. Управление разрешением/запрещением работы буферов осуществляется вручную или от процессора. Пример подобного тестирования на рис. 9.2.

Порт ВВ запрограммирован на вывод, но 8-битный входной порт остается действующим. Данные можно передать на выход от процессора и считать в ЦП по тому же адресу. Внутренние соединения обратной связи между выходными и входными портами реализованы в самой микросхеме ВВ. Линии от микросхемы ВВ подключены к внешнему устройству через неинвертирующий тристабильный 8-битный буфер 74LS244, выходным состояиием которого управляет переключатель $S_1$. Когда $S_1$ заземлен, буфер работает обычным образом; подсоединение $S_1$ к питанию 5 В переводит выходы буфера в высокоимпедансное состояние. Переключатель $S_1$ монтируется на печатной плате, и его положение устанавливается вручную. Нужно обеспечить, чтобы устройство не воспринимало высокоимпедансное состояние как уровень логической «1», что характерно для обычных ТТЛ-схем. Если все выходные линии имеют второй буфер с драйверами Дарлингтона, то такая ситуация не возникает.

### 9.2.2. Инициирование диагностических тестов

В подавляющем большинстве небольших микропроцессорных систем для проверки их основных компонентов применяются диагностические программы, хранимые в ПЗУ. Само ПЗУ может постоянно находиться в системе, а переход к нему осуществляется либо ручным подключением с последующим сбросом компьютера, либо инициированием прерывания, когда ПЗУ размещено в адресном пространстве памяти процедуры обслуживания прерывания. В самых малых системах предусматривается удаление микросхемы системного ПЗУ из панельки, в которую затем вставляется диагностическое ПЗУ.

Диагностическое ПЗУ содержит несколько коротких тест-программ. Обычно ВСЕ программы выполняются как цикл, охватывающий проверки всех устройств, но пользователь может инициировать и любую отдельную тест-программу. Оба варианта широко применяются для тестирования систем. В случае индивидуального выполнения тест-программ их выбор осуществляется через входной порт с подсоединенными к нему переключателями. Микропроцессор считывает состояния переключателей и определяет, какой тест инициировать из диагностического ПЗУ. Входной порт для задания тестов оснащается скользящими переключателями на печатной плате или в качестве него используется порт какого-либо внешнего устройства. Скользящие переключатели удобнее, но их необходимо включать в систему на этапе разработки. К сожалению, тест-процедуры в большинстве систем считаются ненужными излишествами, и специалисты по контролю должны сами как-то освобождать входной порт для целей тестирования.

Мы уже рассматривали самые распространенные виды диагностических тестов для проверки основных компонентов системы. Когда требуется более детальное тестирование периферийных схем и других компонентов, то вызываются соответствующие тесты из внешней памяти. Обычно эти тесты применяются для проверки внешних устройств и для успешной загрузки и исполнения требуют правильного функционирования почти всей вычислительной системы. Следовательно, главное назначение диагностических программ, которые хранятся во внешней памяти, заключается в формировании тест-наборов и последовательностей для проверки периферийных устройств.

# 10. Тестирование функций ввода-вывода

Все рассмотренные в предыдущих главах методы тестирования относились к контролю функций самого компьютера. В общем случае компьютер является только частью системы, которая имеет в своем составе периферийные устройства. Если компьютер функционирует, то его можно использовать для генерирования тестов для периферийных устройств. В большинстве периферийных устройств осуществляется самоконтроль с участием компьютера и показом его результатов на индикаторах «проходит/не проходит». Системы, в которых реализован данный подход, должны быть спроектированы с учетом тестирования, т. е. в них должны быть встроены все дополнительные аппаратные и программные средства. На данном этапе развития компьютерной техники в системах должны быть предусмотрены средства контроля общих функций, а не отдельных устройств, что потребовало бы ненужных и экономически не оправданных затрат разработчиков.

Необходимость учета тестирования при разработке систем связана с требованием введения соответствующих контрольных точек. Для тестирования схемных плат, смонтированных в стойках, следует предусмотреть платы удлинителей. Необходимо также сконструировать удобные средства для подачи электропитания в логический пробник. В документации на систему целесообразно показать типичные формы сигналов, которые можно проверить обычным осциллографом.

Интерфейсы, обеспечивающие взаимодействие компьютера с периферийным устройством или другой системой, проверять довольно затруднительно. Типичными примерами служат параллельная интерфейсная шина IEEE-488 с уникальным протоколом передач и стандарт RS232C последовательной асинхронной передачи. Логические анализаторы рассчитаны на регистрацию данных от систем с шинной структурой, поэтому они практически бесполезны для проверки линии последовательной передачи и не могут оперировать специальными мнемоническими кодами шины IEEE-488. Однако для этих целей используются специализированные анализаторы.

## 10.1. Пример функционального тестирования подсистемы

В любой измерительной системе с компьютером имеется подсистема аналого-цифрового преобразователя, которая часто конструктивно оформляется на одной печатной плате. При считывании электрических аналогов параметров процесса, измеряемых с помощью разнообразных датчиков, сигналы через входной мультиплексор подаются в схему аналого-цифрового преобразователя. Для управления пропорциональными исполнительными органами, например задвижками, необходимы аналоговые выходные сигналы, которые формируют цифро-аналоговые преобразователи. Аналоговая подсистема содержит на одной печатной плате оба вида преобразователей и схемы формирования сигналов.

Аналого-цифровые преобразователи в отдельности проверяются путем подачи на их входы известного эталонного напряжения и выполнения программы, которая каким-либо образом индицирует его цифровой эквивалент. Цифро-аналоговые преобразователи можно проверить с помощью программы, которая посылает в преобразователь определенные двоичные коды, а получающееся выходное напряжение измеряется цифровым вольтметром. Можно подавать также периодические двоичные наборы, пробегающие все состояния, для получения выходного пилообразного сигнала, который наблюдается и измеряется осциллографом.

Под управлением компьютера в аналоговой подсистеме можно осуществить самоконтроль и включить индикатор, если все тесты проходят. С расчетом на этот вариант контроля в схеме необходимо предусмотреть соответствующие подключения, которые в общем виде представлены на рис. 10.1.

Показанная на рис. 10.1 аналоговая подсистема служит примером многочисленных промышленных одноплатных систем сбора данных. Хотя на структурной схеме детали опущены, показанные тракты обратной связи редко реализуются на практике; отсюда следует, что сам компьютер невозможно использовать для самоконтроля подсистемы. Введение относительно несложных схем и программы инициализации при включении питания превращают неконтролируемую систему в систему с самоконтролем. Аналого-цифровой преобразова-

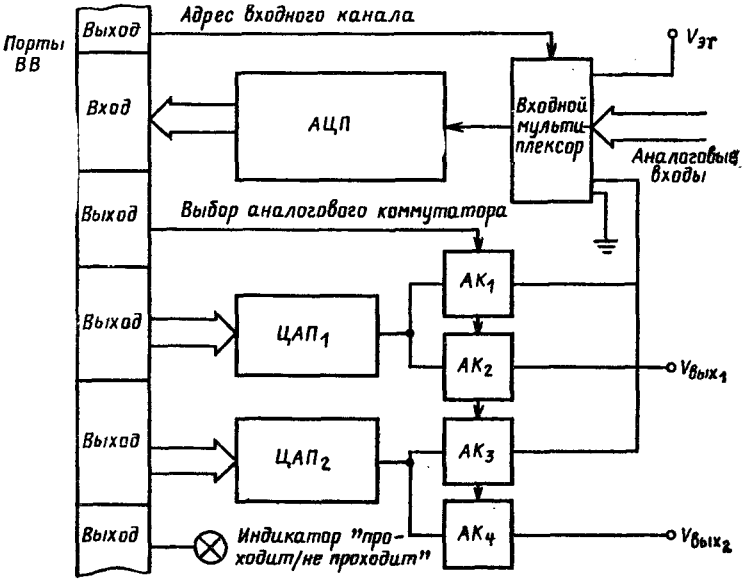



**Рис. 10.1.** Аналоговая подсистема, спроектированная с учетом требований тестирования

тель, показанный на рис. 10.1, можно проверить с любой стороны путем посылки соответствующего адреса на входной мультиплексор и преобразования либо потенциала земли, либо некоторого эталонного напряжения. В данном случае предполагается использование униполярного преобразователя, но можно спроектировать и биполярный преобразователь, если подать еще на один вход мультиплексора отрицательное эталонное напряжение. Современные преобразователи имеют высокое качество, и измерений по принципу «проходит/не проходит» достаточно для проверки состояния аналого-цифрового преобразователя.

На структурной схеме показаны два цифро-аналоговых преобразователя (ЦАП), которые проверяются с помощью аналоговых коммутаторов, передающих их выходные сигналы на вход аналого-цифрового преобразователя *АЦП*. С целью проверки на входы ЦАП периодически подаются известные двоичные наборы, а выходы через *АЦП* преобразуются в цифровые величины, после чего выданный набор сравнивается с полученным. Сле-

довательно, таким способом можно проверить фуикционирование каждого ЦАП, не воздействуя на устройства, к которым ЦАП подключены при обычной работе.

Если каждый основной блок выполняет свои тесты, то включается индикатор «проходит/не проходит», показывая исправность всей аналоговой подсистемы. Можно предусмотреть установку и других индикаторов, например «невозможность калибровки», «отказ $ЦАП_1$», «отказ $ЦАП_2$» и т. п. Рассмотренный способ контроля допускает дальнейшее совершенствование — можно измерить погрешности смещений входных сигналов и учесть их в последующих измерениях переменных. Дальнейшее совершенствование процедур контроля приводит к тому, что еще до выполнения любых измерений система эффективно калибруется путем опроса эталонных напряжений и вычисления смещения для каждого измерения. В предельном случае для контроля *АЦП* во всем диапазоне изменения входных сигналов требуется столько эталонных напряжений, сколько имеется возможных входов, но для современных высококачественных *АЦП* достаточно всего нескольких эталонных точек.

Самоконтроль любых периферийных устройств или подсистем почти всегда требует наличия дополнительных схем, которые необходимо предусмотреть при проектировании. С одной стороны, данный подход оказывается неудовлетворительным, так как дополнительные схемы подвержены отказам, а лишние соединения ухудшают надежность всей системы. С другой стороны, тестирование на функциональном уровне связано с незначительным усложнением схем, но позволяет упростить эксплуатацию системы, в которой главной целью является минимизация простоев. Ремонт осуществляется на уровне неисправных схемных плат путем их замены.

Для функционального тестирования $ЦАП_1$, $ЦАП_2$ и *АЦП* необходимо обеспечить подключение эталонных входных сигналов и возможность управления входным мультиплексором, чтобы компьютер мог выбрать определенный вход и проверить его. Кроме того, необходимо предотвратить подачу потенциально опасных сигналов во внешние схемы при контроле с помощью аналоговых переключателей или тристабильных буферов, а также предусмотреть достаточное число индикаторов «прохо-

дит/не проходит», чтобы персонал мог легко идентифицировать неисправную подсистему. Эти требования тестирования систем предполагают наличие дополнительных портов ВВ для передачи необходимых управляющих сигналов в периферийные устройства.

## 10.2. Тестирование последовательных линий связи

Обычно информация передается между компьютером и периферийными устройствами, например принтерами, видеотерминалами и клавиатурой, в последовательном формате. В своей простейшей форме последовательная передача требует наличия всего двух физических проводников (сигнального тракта и обратной земли), что уменьшает расходы на создание каналов связи. Данные в компьютере хранятся в параллельной форме и для передачи должны быть преобразованы в последовательную форму. Принятые из линии связи данные до обработки в компьютере необходимо преобразовать в параллельную форму. Основой схем преобразования служит регистр сдвига, в который данные загружаются из компьютера в параллельном коде, а выдвигаются последовательно импульсами синхронизации. Регистр сдвига осуществляет и обратное преобразование—данные вводятся из линии связи в последовательном коде, а считываются параллельно.

Имеется несколько стандартов последовательной связи, но самым широко распространенным является стандарт RS232C. Первоначально он был разработан для подключения терминала к *модему (модулятору/демодулятору)*, чтобы обеспечить универсальный способ подсоединения периферийных устройств компьютеров к телефонной линии. Со временем его адаптировали для подключения терминалов к вычислительной системе. Стандарт RS232C предполагает использование 25-контактного разъема, в котором каждый контакт предназначен для сигнала, функция и направление передачи которого полностью определены стандартом. Например, контакт 2 назван XMIT DATA и используется для передачи информации от терминала в модем. Стандарт RS232C определяет также электрические характеристики сигналов — уровни напряжения, максимальную длину линии связи и скорость передачи данных. Перечислим основные характеристики стандарта RS232C:

| | |
|---|---|
| Максимальная длина линии, м . . . | 30 |
| Максимальная скорость, бит/с . . . . | $2\times10^4$ |
| Уровень логической «1», В . . . . . | $-1{,}5\div-36$ |
| Уровень логического «0», В . . . . . | $+1{,}5\div+36$ |

На практике уровни напряжений, соответствующие двум логическим состояниям, равны $\pm12$ В.

Модем должен иметь ответную часть разъема, а терминал — штырьковую. Промышленность сама ввела стандартный 25-контактный разъем *D*-типа.

Описание стандарта RS232C выполнено в терминах модема и терминала. Как классифицировать компьютер в терминах стандарта — как модем или как терминал? В большинстве систем компьютер считается частью связного оборудования (модемом), и на нем устанавливается ответная часть 25-контактного разъема, к которому непосредственно подключается терминал. В том случае, когда доступ к компьютеру осуществляется дистанционно по телефонной линии (рис. 10.2), то возникает проблема взаимного соединения аппаратуры.

Модем и компьютер оснащены ответными частями 25-контактных разъемов *D*-типа, а это означает, что оба конца линии связи предполагают передачу данных через один и тот же контакт. Следовательно, модем попытается передавать данные в компьютер по тому же проводнику, по которому компьютер пытается посылать данные в модем. Кроме того, оба они попытаются принимать данные друг от друга по одной и той же линии. Поэтому взаимодействие между ними невозможно. Проблема соединения аппаратуры решается с использованием «пустого» модема. «Пустой» модем состоит из

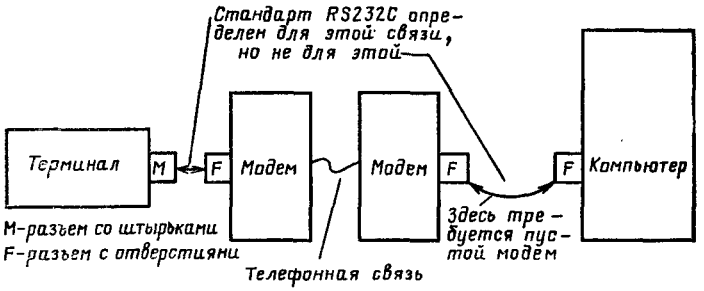



Рис. 10.2. Проблема пустого модема

связного кабеля с одинаковыми разъемами на обоих концах; в данном случае это будут штырьковые части разъемов, но есть «пустой» модем и с ответными частями разъемов. Для обеспечения передачи данных подключение сигнальных линий к контактам на одном конце кабеля изменено так, что с другого конца кабеля они воспринимаются как сигнальные линии от оборудования другого типа. Таким образом, линии передаваемых данных и линии принимаемых данных упорядочиваются по номерам контактов на разъеме вместе со своими управляющими сигналами.

Стандарт RS232C определяет 20 из 25 контактов разъема и предусматривает два канала передачи. Применяются следующие мнемоники сигналов:

AB — сигнальная земля;
CE — индикатор звонка (из модема);
CD — готовность терминала DTR (в модем);
CC — готовность модема DSR (из модема);
BA — передаваемые данные( в модем);
BB — принимаемые данные (из модема);
DA — синхронизация передатчика (в модем);
DB — синхронизация передатчика (из модема);
DD — синхронизация приемника (из модема);
CA — запрос передачи RTS (в модем);
CB — сброс передачи CTS (из модема);
CF — детектор сигнала приемной линии (из модема);
CG — детектор качества сигнала (из модема);
CH — селектор скорости передачи данных (в модем);
CI — селектор скорости передачи данных (из модема);
SBA — вторичные передаваемые данные;
SBB —вторичные принимаемые данные;
SCA — вторичный запрос передачи;
SCB — вторичный сброс передачи;
SCF — детектор сигнала вторичной приемной линии.

На практике вторичный канал применяется редко. Часто в связном канале между компьютером и терминалом для управления передачей информации применяются только 4 линии квитирования — RTS, CTS, DTR и DSR. В простейшем интерфейсе для передающего/принимающего терминала используются только линии сигнальной земли (AB), передаваемых данных (BA) и принимаемых данных (BB). Однако периферийное оборудование на одном конце линии может требовать управляющий сигнала CTS, который разрешает ему пе-

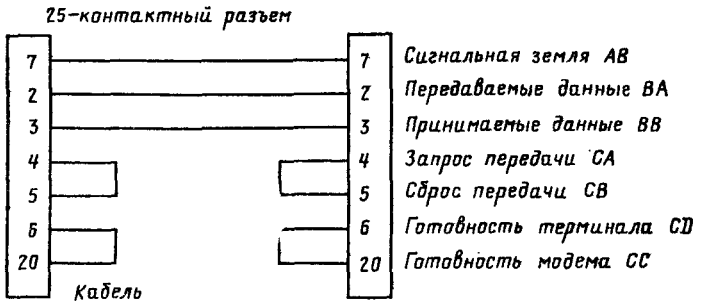



**Рис. 10.3.** Соединения автоответа

редавать данные. Если этот сигнал не формируется системой на противоположном конце кабеля, то его необходимо аппаратно встроить в разъем, чтобы устройство как бы принимало сигнал квитирования с другого конца. Такое подключение осуществляется в обоих разъемах и называется автоответом. Рассмотренные простейшие взаимосвязи стандарта RS232C показаны на рис. 10.3.

Соединения автоответа можно реализовать и в пустом модеме. Они отличаются от приведенных на рис. 10.3 тем, что линии, подсоединенные к контактам *2* и *3*, меняются местами на одном конце кабеля, при этом оба разъема одинаковы (оба имеют либо ответные, либо штырьковые части).

#### 10.2.1. Протоколы последовательной передачи

Стандарт RS232C обеспечивает общие средства реализации связного канала между компьютером и периферийными устройствами. Однако прежде чем передавать информацию между двумя системами, обе они должны быть запрограммированы на одинаковые формат и скорость передачи данных.

Наиболее распространенным способом кодирования, который обеспечивает передачу символьных и управляющих кодов между системами, служит 7-битный код ASCII (Американский стандартный код для обмена информацией). Именно он почти всегда применяется для передачи данных по последовательным линиям.

Если выбран код ASCII и реализован стандарт

RS232C, остается определить порядок передачи информации. Этот порядок называется протоколом передачи, и он определяет не только порядок передачи кодов символов, но и порядок работы с управляющими битами, которые необходимы для синхронизации взаимодействующих систем. Приемнику необходимо сообщить о начале и об окончании передачи символа, кроме того, требуется передавать контрольные биты для обнаружения ошибок при передаче. На практике применяются два основных способа передачи — синхронная передача и асинхронная передача. Первый из них применяется для передачи блоков символов, а второй — для передачи одиночных символов. Протоколы синхронной передачи намного сложнее протоколов асинхронной передачи и не охватываются стандартом RS232C.

При асинхронной передаче требуется, чтобы от передатчика в приемник первым был послан *стартовый бит,* информирующий о начале передачи символа. После этого передается 7-битный код символа, причем первым передается младший бит, а последним — старший бит. Вслед за 7 битами символа передается контрольный бит или бит *паритета.* Способ реализации бита паритета в различных системах варьируется. Бит паритета может отсутствовать совсем, но может применяться четный/нечетный паритет. Если, например, выбран четный паритет, передатчик подсчитывает количество единиц в передаваемом коде символа и, когда в результате получается четное число, устанавливает в бите паритета уровень логического 0. Когда же после суммирования получается нечетное число, бит паритета устанавливается передатчиком в состояние логической 1, так что воспринимаемое приемником общее число единиц между стартовым и стоповым битами всегда является четным. Нечетный паритет используется аналогичным образом. Единственный контрольный бит паритета позволяет обнаруживать все однобитные ошибки передачи, но может не зафиксировать многобитные ошибки. Приемник подсчитывает число принятых состояний логической 1 и фиксирует ошибку, если результат не совпадает с используемым типом паритета. После передачи бита паритета до начала передачи следующего символа линия передачи остается незадействованной в течение одного, полутора или двух периодов синхронизации. Эти стоповые биты сохранились с того времени, когда периферийные устройства

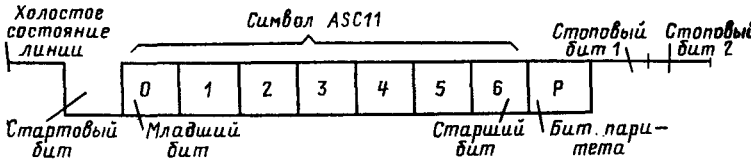



**Рис. 10.4.** Протокол асинхронной передачи

в своем большинстве были электромеханического типа и не могли мгновенно реагировать на передачу следующего символа. Формат асинхронной передачи данных представлен на рис. 10.4.

До передачи любого символа в линии присутствует уровень +12 В, который в стандарте RS232C считается состоянием логического 0. Если в результате какого-нибудь повреждения (или отключения на любом конце) целостность линии нарушается, ее потенциал будет нулевым, и это состояние приемник фиксирует как неисправность линии.

Передатчик посылает информацию в линию, пользуясь своими сигналами синхронизации, а приемник воспринимает ее, используя собственные сигналы синхронизации, частота которых обычно в 16 раз больше. Первый отрицательный переход в линии считается стартовым битом, после чего приемник отсчитывает 8 своих импульсов синхронизации и опрашивает состояние линии. Опрос приходится как раз на середину стартового импульса. Если приемник фиксирует низкий уровень напряжения, он считает сигнал действительным стартовым импульсом; в противном же случае идентифицируется помеха и приемник переходит в состояние ожидания действительного фронта стартового импульса.

Если обнаружен действительный стартовый импульс, приемник отсчитывает последовательности из 16 импульсов своей собственной синхронизации, что позволяет ему опрашивать состояние линии посередине периода передачи каждого бита символа и бита паритета. Получаемые при этом биты передаются в регистр сдвига, и после его заполнения символ параллельным кодом передается в вычислительную систему. Пусть, например, передается прописная буква А, имеющая в коде ASCII 16-ричное представление 41. Когда используется четный паритет, бит паритета устанавливается в состояние ло-

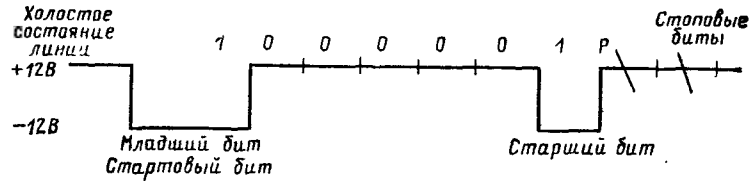



**Рис. 10.5.** Код ASCII буквы А в последовательной линии

гического 0, чтобы общее число бит с логической 1 было четным. Двоичное представление буквы А в 7-битном коде имеет вид 1000001, а с учетом бита паритета и обратного порядка передачи получается последовательность 10000010.

Информация в последовательной линии связи представляется в формате *без возврата к нулю,* т. е. между битами линия не возвращается к состоянию логического 0. Если, например, два соседних бита находятся в состояниях логической 1, линия остается в этом состоянии в течение двух периодов. На экране осциллографа передаваемый по линии символ А с четным паритетом имеет временную диаграмму, показанную на рис. 10.5.

Чтобы не допустить перекоса синхронизации, частоты сигналов синхронизации передатчика и приемника не должны отличаться более чем на 2 %.

#### 10.2.2. Скорость передачи

Скорость передачи символов измеряется количеством бит, передаваемых в секунду. Эта величина называется *бод.* Скорость передачи в бодах относится к общему числу бит, передаваемых в символе, включая стартовый и стоповый биты, а также бит паритета. Обычно 7-битный символ в коде ASCII сцепляется с одним стартовым битом, двумя стоповыми битами и одним битом паритета, что дает всего 11 бит. Если, например, скорость передачи в бодах составляет 110, то с учетом 11 бит на символ по каналу связи за одну секунду можно передать 10 символов. Такая низкая скорость передачи характерна для старых электромеханических телетайпов. В более современных системах скорость передачи доходит до 9600 бод, что позволяет передать за секунду 872 символа (по-прежнему предполагая 11 бит в символе).

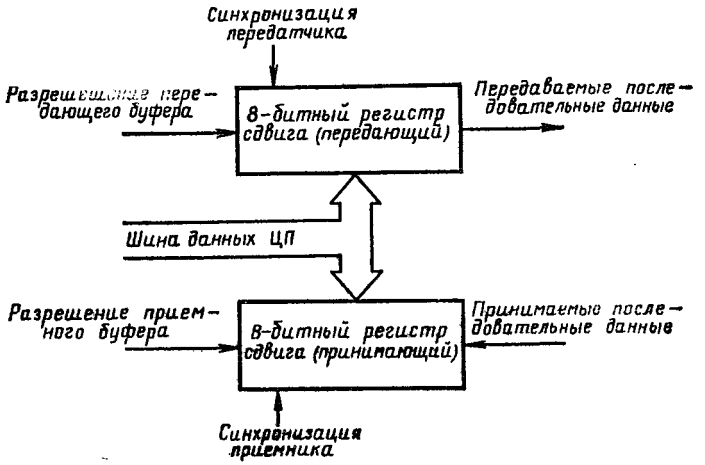



**Рис. 10.6.** Процесс параллельно последовательного преобразования

### 10.2.3. Асинхронный связной интерфейс

В преобразованиях между параллельным форматом информации в вычислительной системе и последовательным форматом асинхронной передачи важнейшими компонентами оказываются два регистра сдвига. Один из них обрабатывает данные от вычислительной системы, а второй принимает данные, которые затем считываются системой. В общем виде этот способ передачи представлен на рис. 10.6.

Данные от ЦП загружаются в *передающий регистр*, а затем последовательно выдвигаются из него со скоростью, определяемой *синхронизацией передатчика*. Принимаемые данные вдвигаются в принимающий регистр со скоростью, определяемой *синхронизацией приемника*, и при заполнении регистра его содержимое считывается в шину данных. В этой простой схеме, показывающей основные функции, отсутствуют несколько важных элементов. Необходимо ввести дополнительные схемы, фиксирующие заполнение принимающего регистра и проверяющие паритет. Кроме того, необходимо удалить служебные биты, чтобы ЦП считывал только биты символа. В передающей секции требуются схемы генерирования стартового бита и бита паритета, которые до передачи слова добавляются к битам символа. Все эти

средства реализованы в одной БИС, называемой *универсальным асинхронным приемником/передатчиком* (УАПП).

УАПП было одно из первых БИС с массовым выпуском. Первые микросхемы программировались по таким параметрам, как четный/нечетный паритет и число стоповых бит, путем подключения соответствующих выводов БИС к линии электропитания. Последующие микросхемы УАПП настраивались на режим программой и были рассчитаны на прямое подключение к шине данных ЦП. В них были предусмотрены основные управляющие сигналы стандарта RS232C, а типичным примером служит УСАПП 8251A фирмы Intel*. Эта полностью программируемая микросхема может реализовать протоколы синхронного обмена информацией, например SDLC. Конструктивно микросхема 8251A выполнена в корпусе БИС с 28 внешними выводами и содержит все схемы, необходимые для генерирования стартового и стоповых бит, а также бита паритета. Ее структурная схема представлена на рис. 10.7.

Принятые данные можно считать из микросхемы по сигналам низкого уровня на управляющих линиях $\overline{CS}$ (выбор кристалла) и $\overline{RD}$ (считывание). Передаваемые данные загружаются в УСАПП по сигналам низкого уровня на управляющих линиях $\overline{CS}$ и $\overline{WR}$ (запись). Для программиста микросхема состоит из четырех регистров, в два из которых можно записывать, а с двух можно считывать. Микросхема программируется (принимает управляющую информацию из шины данных) по сигналу логической 1 на линии $C/\overline{D}$ (управление/данные), а при загрузке передаваемых данных на линии $C/\overline{D}$ действует уровень логического 0.

С каналом последовательной связи микросхема соединена 11 линиями, включая отдельные линии синхронизации для передающей и приемной секций. Во многих случаях скорости передачи и приема данных различны. В интерфейсе видеодисплея принимаемые данные появляются с относительно малой скоростью, так как они формируются при ручном вводе с клавиатуры; в то же время передаваемые данные можно отобразить на экра-

* УСАПП — универсальный синхронно-асинхронный приемник/передатчик. — *Прим. ред.*

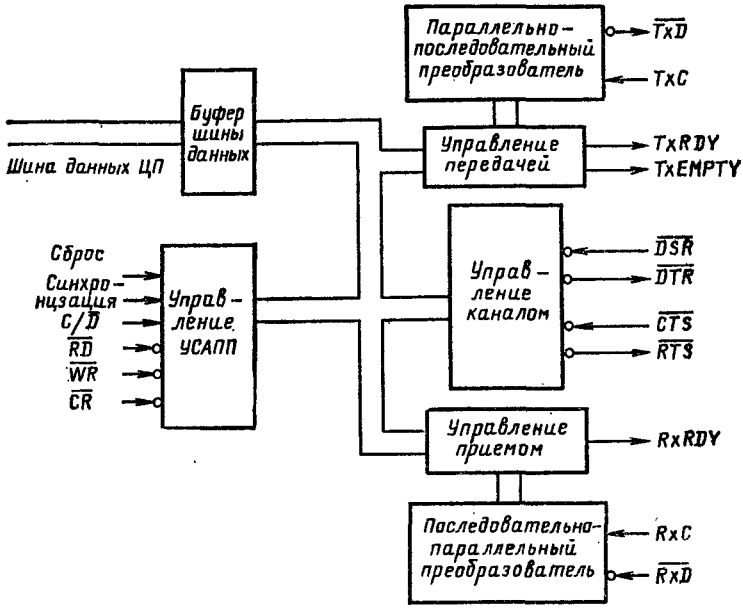



**Рис. 10.7.** Блок-схема УСАПП 8251А

не с гораздо более высокой скоростью. Выдаваемые из УСАПП последовательные данные появляются на линии $\overline{TxD}$, а принимаемые данные от некоторого удаленного внешнего устройства передаются в УСАПП по линии $\overline{RxD}$. Данные выдвигаются из передающего регистра сигналами синхронизации на линии *TxC*, а вдвигаются в принимающий регистр сигналами на линии *RxC*. Передающая и приемная секции имеют двухтактные буферные регистры. Посылаемые из ЦП данные вначале загружаются в буферный регистр передатчика, который сигналом высокого уровня на выходе *TxRDY* информирует ЦП о своей готовности принять новые данные. Уровень сигнала на этой линии ЦП считывает как один из бит в регистре состояния, содержимое которого считывается из микросхемы УСАПП при высоком уровне сигнала $C/\overline{D}$. Когда передающий регистр оказывается пустым, на линии *TxEMPTY* формируется сигнал высокого уровня и содержимое буферного регистра загружается в передающий регистр для последующей передачи.

Аналогично принятые данные до считывания центральным процессором загружаются в буферный регистр, состояние которого отражается сигналом на выходной линии *RxRDY*. При считывании данных из буфера сигнал *RxRDY* сбрасывается. Данные из УСАПП считываются при низком уровне сигнала $C/\overline{D}$.

Частота синхронизации приемника должна быть в 16 раз выше частоты поступления бит входных данных. В этом случае биты символа опрашиваются почти в середине временного интервала их представления, что допускает некоторый перекос синхронизации на приемном и передающем концах линии связи. Если установлена скорость передачи 1200 бод и символ содержит 11 бит, продолжительность передаваемого бита составляет 833 мкс. Частота сигнала *RxC* должна быть в 16 раз выше, и период синхронизации оказывается равным 52 мкс. Частота сигнала синхронизации на входе *TxC* также должна быть в 16 раз выше заданной скорости передачи, поэтому при работе передающей секции со скоростью 1200 бод линию *TxC* можно подключить к линии *RxC*. Для внутренних схем УСАПП 8251А необходима синхронизация с частотой 1 МГц, которая обычно получается от генератора системной синхронизации. Если ЦП работает с частотой 1 МГц, период синхронизации равен 1 мкс; тогда общий сигнал для входов *TxC* и *RxC* можно получить из системной синхронизации путем деления на 52. Это число разлагается на множители 13 и 4, а получение сигналов синхронизации *TxC* и

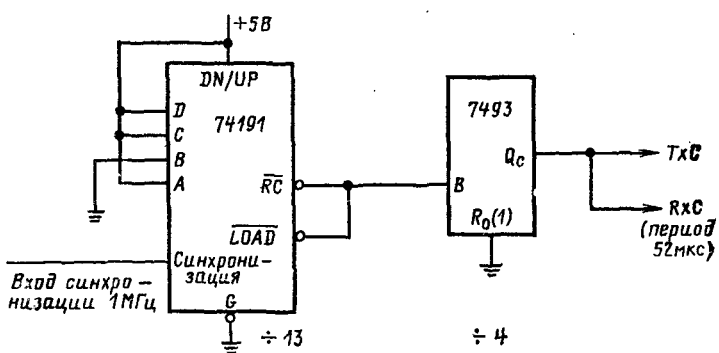



**Рис. 10.8.** Генерирование синхронизации

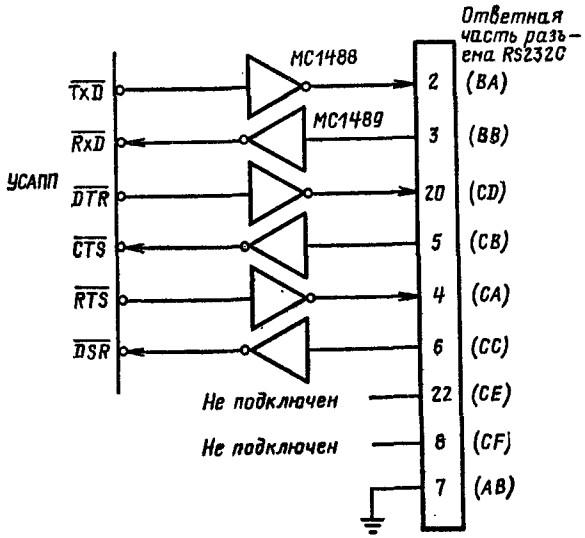



**Рис. 10.9.** Интерфейс RS232C

*RxC* обеспечивает схема, приведенная на рис. 10.8 и состоящая из двух счетчиков по модулю 13 и 4.

Уровни сигналов, передаваемых и принимаемых УСАПП, являются ТТЛ-совместимыми. Однако уровни сигналов в интерфейсе RS232C составляют ±12 В; следовательно, в системе требуются схемы преобразователей уровней. В настоящее время выпускаются ИС, выполняющие функции преобразователей уровней; наиболее распространены микросхема MC1488 для преобразования ТТЛ-сигналов в сигналы с уровнями ±12 В и микросхема MC1489 для преобразования сигналов с уровнями ±12 В в ТТЛ-сигналы. Типичное подключение УСАПП к линиям последовательной связи с использованием этих микросхем показано на рис. 10.9.

Для проверки последовательного интерфейса RS232C необходим анализатор, который можно программировать на скорость передачи в бодах и который может регистрировать и индицировать данные, желательно в коде ASCII. Анализатор регистрирует информацию посредством восприятия стартового бита и последующего опроса линии для определения каждого передаваемого бита. УСАПП может реализовать эту функцию и преоб-

разовать каждый последовательно передаваемый символ в параллельное слово, записываемое во внутреннее ЗУПВ анализатора. После заполнения ЗУПВ встроенный в прибор микропроцессор дешифрирует хранящиеся двоичные коды и выводит на экран их символьные эквиваленты.

В первых анализаторах последовательных линий после задания скорости передачи и типа паритета производились регистрация информации и преобразование данных в параллельный формат для индикации на экране осциллографа. Именно в такой ограниченной форме осуществляется отображение информации в анализаторе последовательных кодов 1620A фирмы Hewlett-Packard. В более поздних анализаторах, например в анализаторе последовательной линии 1640B той же фирмы, данные индицируются на экране в формате ASCII. В пассивном режиме эти индикаторы только регистрируют и индицируют информацию, а в активном режиме моделируют терминал на линии RS232C и осуществляют передачу и прием данных через последовательный интерфейс компьютера.

### 10.3. Контроль приборной шины IEEE-488

Скорость передачи данных между системами резко увеличивается при переходе от последовательного формата к параллельному. Однако по мере увеличения линий в шине быстро растут сложность схем интерфейса и расходы на соединительные кабели. Компромиссный вариант заключается в том, чтобы зарезервировать для шины данных, например, 8 линий и ввести несколько линий для управления передачами по шине. Такой подход был предложен фирмой Hewlett-Packard для объединения испытательного оборудования и реализован в виде интерфейсной шины фирмы Hewlett-Packard. Шина получила широкое распространение и имеет несколько названий: универсальная интерфейсная шина (GPIB); стандарт IEEE-488 1975 г., который был преобразован в стандарт IEEE-488 1978 г.; шина ANSI (стандарт MC1.1-1975 Американского национального института стандартов); шина IEC 625-1 (стандарт Международной электротехнической комиссии).

Стандарт GPIB на параллельную передачу данных устанавливает спецификации на электрические и меха-

нические параметры шины; уровни сигналов ТТЛ-совместимы, но требования по току превышают возможности стандартных ТТЛ-микросхем. Однако выпускаются микросхемы, удовлетворяющие всем спецификациям стандарта.

Скорость передачи данных до 1 Мбит/с, и к шине одновременно может быть подключено до 15 основных устройств. Под основным устройством здесь понимается законченный прибор, например программируемый цифровой вольтметр, генератор функций или накопитель на гибком диске. Внутри каждого основного устройства разрешены обращения максимум по 32 вторичным адресам. Один из вторичных адресов может, например, относиться к секции вольтметра постоянного тока цифрового мультиметра, а другой — к секции омметра.

Физически подключенные к шине устройства должны размещаться на удалении не более чем 4 м при максимальной длине шины, равной 20 м. Длина шины передачи определяется как удвоенное число устройств на шине, представленное в метрах. Например, если к шине подключены всего три устройства, максимальная длина передачи сокращается до 6 м. Стандарт IEEE-488 определяет 24-контактный разъем. Разъемы позволяют реализовать звездную или радиальную конфигурацию системы связи устройств. Отметим, что Европейский стандарт IEC 625-1 определяет 25-контактный разъем, и для подключения к шине IEEE-488 приборов, рассчитанных на этот формат, требуются специальные адаптеры.

Шина предназначена для объединения программируемых приборов и функционирует под управлением контроллера шины, который управляет всеми передачами данных. Передаваемые данные не проходят через контроллер, но могут быть выданы на шину одним прибором и восприняты несколькими другими под общим управлением контроллера.

#### 10.3.1. Структура универсальной интерфейсной шины

В универсальной интерфейсной шине применяются 16 линий: 8 для передаваемых данных и еще 8 для сигналов управления. Эти 16 линий можно объединить в три группы: линии ввода-вывода данных (8 линий), линии управления передачами (3 линии) и линии управления

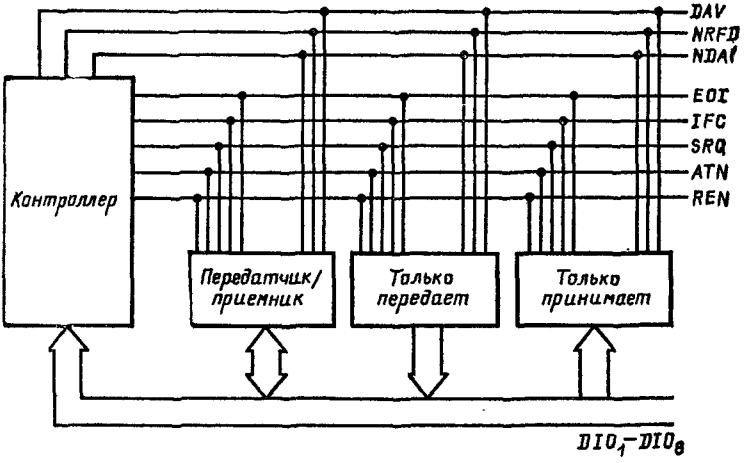



**Рис. 10.10.** Системные подключения универсальной интерфейсной шины

интерфейсом (5 линий). Каждая линия имеет мнемоническое обозначение:

| | | |
|---|---|---|
| *DI/O*$_1$ — *DI/O*$_8$ | — линии передач. данных, | |
| *DAV* | — данные действительны, | Линии управления передачами |
| *NRFD* | — неготовность данных, | |
| *NDAC* | — даиные ие восприняты, | |
| *EOI* | — конец или идентификация, | Линии управления интерфейсом |
| *IFC* | — сброс иитерфейса, | |
| *SRQ* | — запрос обслуживания, | |
| *ATN* | — внимание, | |
| *REN* | — дистанционное разрешение | |

Общая структура шины приведена на рис. 10.10.

Система может иметь только один контроллер, который управляет передачей данных из одного в один или в несколько других приборов. Любой подключенный к шине прибор может быть *передатчиком* (talker) или *приемником* (listener). Передатчик может только выдавать данные на шину под управлением контроллера, а приемник может только восприиимать данные с шины также под управлением контроллера. Любой прибор в каждый момент времени может быть только передатчиком или только приемником, но контроллер может ре-

программировать прибор и изменить его функцию. Линии $DI/O_1$—$DI/O_8$ применяются не только для передачи данных из одного прибора в другой, но и для передачи адресной информации. Прибор программируется как передатчик или приемник адресным двоичным кодом, посланным ему контроллером.

Скоростью передачи данных от передатчика к приемникам управляют три линии квитирования *DAV, NRFD* и *NDAC*. Линии управления интерфейсом осуществляют передачу управляющей информации, которая может быть инициирована контроллером или передатчиком путем выдачи сообщения на шину данных.

*EOI* — конец или идентификация. На этой линии контроллер или передатчик формирует истинный (низкий) уровень, когда по шине передан последний байт сообщения, т. е. этот сигнал означает конец передачи. Совместно с линией *ATN* применяется для инициирования последовательности опроса с целью определения наличия запроса на обслуживание от устройств, подключенных к шине.

*IFC* — сброс интерфейса. На этой линии истинный (низкий) уровень может сформировать только контроллер; этот сигнал переводит все подключенные к шине устройства в нерабочее состояние. По существу, он действует как сигнал общего сброса.

*SRQ* — запрос обслуживания. На линии *SRQ* истинный (низкий) уровень может сформировать любое подключенное к шине устройство, которое требует обслуживания от контроллера; фактически он действует как сигнал прерывания программы контроллера. Например, этот сигнал может выдать цифровой вольтметр, когда в ием образовано новое показание. По сигналу *SRQ* контроллер с помощью сигналов в линиях *EOI* и *ATN* переходит в режим опроса, чтобы определить, какое устройство запрашивает обслуживание.

*ATN* — внимание. Сигнал в эту линию может выдать только системный контроллер; активный (низкий) уровень на линии *ATN* информирует все устройства о том, что на шине *DI/O* находится адресная или управляющая информация. Когда сигнал на линии *ATN* возвращается к высокому уровню (идентифицируя наличие данных на линиях *DI/O*), взаимодействие с шиной разрешается только тем устройствам, которые ранее были определены как передатчики или приемники.

*REN* — дистанционное разрешение. Активный сигнал (низкого уровня) на эту линию может выдать только системный контроллер, и он должен сохраняться таковым во время любой передачи по шине. Если на линии *REN* появляется высокий уровень, то все устройства, подключенные к шине, переводятся в режим локального управления. Предположим, что генератор функций вручную запрограммирован с пульта управления на формирование треугольного сигнала с заданными периодом и амплитудой и что прибор был репрограммирован через шину как генератор синусоидального сигнала. Тогда, если контроллер установит на линии *REN* высокий уровень, выход генератора изменится с синусоидального сигнала на треугольный сигнал.

Иногда, если линия запроса прерывания *SRQ* в системе не задействована, линия *REN* постоянно подключена к земле, чтобы предотвратить переход устройств в режим локального управления; в этих случаях управление в основном осуществляется по линиям *ATN* и *EOI*.

Во всех передачах данных участвуют три линии управления передачами *DAV, NFRD* и *NDAC*. Реализуемый ими протокол квитирования обеспечивает режим передачи, при котором данные не снимаются с шины до тех пор, пока их не воспримут и не запомнят все устройства, запрограммированные на получение этих данных. Все передачи данных осуществляются со скоростью, определяемой самым медленным устройством, подключенным к шине. Упрощенная временная диаграмма квитирования с участием рассматриваемых линий показана на рис. 10.11.

Линией *DAV* управляет передатчик, а линиями *NRFD* и *NDAC* — активизированные приемники на шине. Сиг-

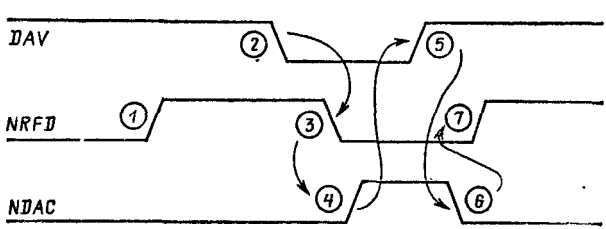



**Рис. 10.11.** Упрощенная процедура квитирования универсальной интерфейсной шины

нал *DAV* показывает наличие на шине действительных данных; сигнал *NRFD* указывает на готовность приемников к приему данных, и сигнал *NDAC* идентифицирует окончание приема данных приемниками.

Предположим, что контроллер определил передатчик и запрограммировал все приемники. Тогда все передатчики формируют сигналы высокого уровня на линии *NRFD*, как показано в точке *1* на рис. 10.11. Передатчик контролирует линию *NRFD* и при появлении в ней сигнала высокого уровня определяет, что все приемники готовы принимать данные, которые он выдаст на шину. После этого передатчик помещает данные на шину и отводит небольшой временно́й интервал на установление новых состояний в линиях шины. По истечении этого интервала передатчик формирует сигнал низкого уровня на линии *DAV*, сообщая всем приемникам о том, что данные на шине действительны. Эта ситуация представлена на рис. 10.11 точкой *2*.

Все приемники фиксируют сигнал низкого уровня на линии *DAV* и формируют сигнал низкого уровня на линии *NRFD*, когда они готовы принимать данные. После того как каждый приемник загрузит данные во внутренний буфер, он освобождает линию *NDAC*, указывая на окончание приема данных. Процесс передачи может продолжиться только после того, как линию *NDAC* освободит самый медленнодействующий приемник. Это показано на рис. 10.11 точкой *4*.

Передатчик обнаруживает сигнал высокого уровня на линии *NDAC* и формирует такой же сигнал на линии *DAV*, показывая, что данные на шине больше недействительны. В свою очередь каждый приемник обнаруживает сигнал высокого уровня на линии *DAV* и формирует сигнал низкого уровня на линии *NDAC*, подтверждая, что данные с шины сняты (см. точку *6* на рис. 10.11). После этого каждый приемник формирует на линии *NRFD* сигнал высокого уровня, показывая тем самым свою готовность принимать следующий байт данных.

Последовательность передачи на этом завершена, и все приемники ожидают следующий байт данных. Эта последовательность составляет суть всех передач по шине и учитывает задержки распространения сигналов и время обработки информации в приемниках.

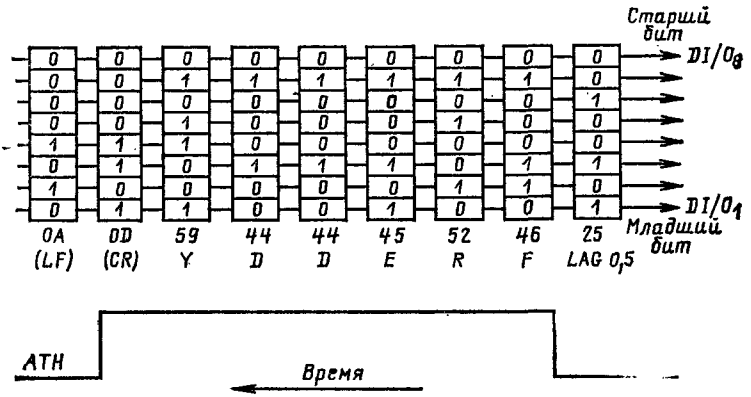



Рис. 10.12. Передача слова *FREDDY* по универсальной интерфейсной шине

### 10.3.2. Пример передачи по шине

Предположим, что по универсальной интерфейсной шине необходимо передать слово *FREDDY* в приемник с первичным адресом 05. Будем считать, что контроллер уже послал целевой адрес в передающий прибор, что разрешает ему выдавать данные на шину. Передача по шине осуществляется в соответствии с рис. 10.12.

Символы передаются по шине в коде ASCII, когда на линии *ATN* действует сигнал высокого уровня, сопровождающий передачу данных. В примере на рис. 10.12 предполагается, что слово *FREDDY* передается в один приемник, первичный адрес которого равен 05. При низком уровне на линии *ATN* по шине передается адресная информация; первая передача программирует устройство 05 на работу в качестве приемника. Три старших бита в передаче адреса информируют указанный прибор о том, что он должен действовать как передатчик или приемник. В данном случае эти три бита содержат код 001, и устройство 05 настраивается на режим приема. При программировании устройства в качестве передатчика эти биты содержат код 010. Первичные и вторичные адреса сведены в группы, которым присвоены мнемонические коды; код данной конкретной передачи *LAG* 05 показывает, что он принадлежит группе первичных адресов приемников. Младшие пять бит адресного байта задают первичный адрес устройства (в нашем случае он равен 05).

После этого по шине пересылается каждый символ слова в коде ASCII. Передача сопровождается сигналом высокого уровня на управляющей линии *ATN*. В конце передачи формируются коды символов *возврата каретки* (*CR*) и *перевода строки* (*LF*). Во время передачи символа перевода строки на линии *EOI* формируется сигнал низкого уровня, идентифицирующий последний передаваемый байт.

При передаче по шине любого адреса или символа данных инициируется протокол квитирования, показанный на рис. 10.11. Поэтому в примере, представленном на рис. 10.12, будет девять таких последовательностей квитирования.

Такой прибор, как цифровой вольтметр в основном работает в качестве передатчика и посылает данные по шине в приемники и (или) контроллер. Для программирования цифрового вольтметра контроллер должен адресовать его как приемник, а затем посылать цепочку управляющих символов, определяющих его режим работы. Цепочка символов в коде ASCII может, например, запрограммировать его на измерение постоянного напряжения с автоматическим выбором диапазона и включенным входным фильтром для сглаживания входных измеряемых сигналов. Такая конкретная цепочка может иметь вид FOROJ1, где первая пара FO выбирает функцию вольтметра постоянного напряжения, пара RO определяет режим автоматического выбора диапазона, а последняя пара J1 подключает входной фильтр. Каждый прибор, который допускает программирование по приборной шине, содержит в руководстве описание всех режимов работы, на которые он может быть настроен путем загрузки в него управляющих символов из контроллера шины.

### 10.3.3. Анализ сигналов универсальной интерфейсной шины

Данные можно вводить в логический анализатор по спадающему или нарастающему фронту сигнала на линии *DAV*, по спадающему фронту сигнала на линии *NRFD* или по нарастающему фронту сигнала на линии *NDAC*. Синхронизация данных по спадающему фронту сигналов *DAV* или *NRFD* вызывает регистрацию данных до их передачи, а нарастающий фронт сигналов *DAV* или *NDAC* обеспечивает фиксацию и прослеживание

данных только после того, как они восприняты всеми приемниками.

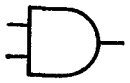
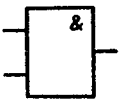
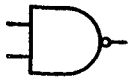
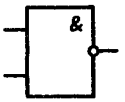
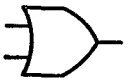
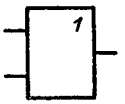
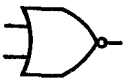
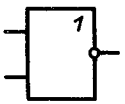
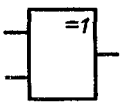
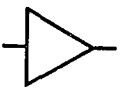
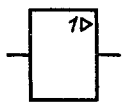
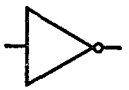
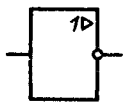
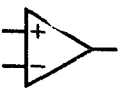
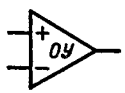
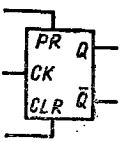
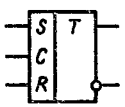
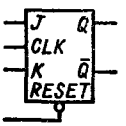
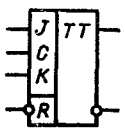
Имеется несколько анализаторов, с помощью которых можно исследовать передачи по приборной шине. Анализатор 1602A фирмы Hewlett-Packard обеспечивает запоминание до 64 передач по шине с последующей индикацией в одном из нескольких форматов (включая код ASCII) на однострочном индикаторе. Логический анализатор 7D01 фирмы Tektronix, который вставляется в осциллографы серии 7000, с форматтером данных DF2 обеспечивает индикацию мнемоник приборной шины вместе с линиями управления интерфейсом. Типичная строка имеет вид

*ATN* *LAG* 18 *REN* 1011

Такая запись означает, что на линии внимания действует сигнал низкого уровня, идентифицирующий наличие на шине адресной информации . Символы *LAG*18 означают, что устройство с первичным адресом 08 программируется как приемник. Далее показано наличие активного сигнала *REN* — все подключенные к шине устройства работают под управлением контроллера. Последние четыре цифры предназначены для индикации состояний четырех отдельных входов в анализатор, которые используются для контроля других сигналов в системе.

Анализатор 7D01 с форматтером DF2 индицирует на экране осциллографа одновременно до 17 передач по шине, но во внутренней памяти анализатора может храниться до 254 отпечатков передач, и их можно вывести на экран в режиме «рулона». Анализатор может работать в режиме временны́х диаграмм, что обеспечивает исследование таких отказов, которые приводят к неправильным последовательностям квитирования.

**Обозначения элементов электронных схем**

| В книге | По ГОСТ | Функциональное назначение элемента |
|---|---|---|
|  |  | Логическое И (AND) |
|  |  | Схема И—НЕ (NAND) |
|  |  | Логическое ИЛИ (OR) |
|  |  | Схема ИЛИ—НЕ (NOR) |
| |  | Исключающее ИЛИ (XOR) |
|  |  | Усилитель мощности |
|  |  | Инвертирующий усилитель |
|  |  | Операционный усилитель |
|  |  | RS-триггер |
|  |  | JK-триггер |

## СПИСОК ЛИТЕРАТУРЫ


### Общие вопросы поиска неисправностей в цифровых схемах

1. **Techniques** of digital troubleshooting. Hewlett-Packard Application Note 163-1, July 1973.
2. **The IC** troubleshooters. Hewlett-Packard Application Note 163-2.
3. **Slater M., Bronson B.** Practical microprocessors. Hewlett-Packard, March 1979.

### Логические анализаторы

1. **Farley B.** Logic analyzers. Digital Design, January 1980.
2. **Staff** of Hewlett-Packard. Logic analyzers. Digital Design, December 1978.
3. **Rickman S. A.** Selecting a logic state analyzer. Digital Design, June 1979.
4. **Mapping**, a dynamic display of digital system operation. Hewlett-Packard Application Note 167-6.
5. **Engineering** in the data domain cells for a new kind of digital instrument. Hewlett-Packard Application Note 167-4.
6. **The role** of logic state analyzers in microprocessor based designs. Hewlett-Rackard Application Note 167-13.
7. **Brampton J.** State analyzers move from lab to production area. Electronic Design, May 1982.

### Сигнатурный анализ

1. **Hewlett-Packard** Journal, October 1979.
2. **Hewlett-Packard** Journal, May 1977.
3. **Palley L.** $E^2$ PROMs bring flexibility to in-system signature analysis. Electronic Product Design, February 1983.
4. **Peacock D.** Signature analysis aids production testing. New Electronics, December 1979.
5. **Beers R.** Dual-mode signature analysis aids instrument trouble — shooting. EDN, April 1982.
6. **An introduction** to data compression measurement for digital logic. Data brief number 301, Solartron Instrumentation Group.

### Тестирование периферийных устройств

1. **Using** the 1620 A for serial pattern recognition. Hewlett-Packard Application Note 167-10.
2. **Monitoring** the IEEE-488 dus with the 1602A logic state analyzer. Hewlett-Packard Application Note 280-2.

## ОГЛАВЛЕНИЕ


Предисловие к русскому изданию . . . . . 3
Предисловие . . . . . 5
Список условных обозначений . . . . . 10

**1. Введение в микропроцессорные системы** . . . . . 11
1.1. Компьютер . . . . . 12
1.2. Системы с шинной структурой . . . . . 13
1.3. Системы с отображением ввода-вывода на память . 16
1.4. Тристабильные схемы . . . . . 17
1.5. Дешифрирование адреса . . . . . 20
1.6. Микроконтроллер . . . . . 28
1.7. Уровни программирования . . . . . 30

**2. Проблемы тестирования систем** . . . . . 36
2.1. Аппаратные средства или программное обеспечение? 37
2.2. Передачи параллельных кодов последовательно во времени . . . . . 38
2.3. Мультиплексирование шины . . . . . 39
2.4. Проблемы тестирования микросхем . . . . . 39
2.5. Системное ядро . . . . . 41
2.6. Тестирование ЦП . . . . . 43
2.7. Тестирование ПЗУ . . . . . 44
2.8. Тестирование ЗУПВ . . . . . 47
2.9. Тестирование ввода-вывода . . . . . 50
2.10. Некоторые общесистемные проблемы . . . . . 52
2.10.1. Блоки электропитания . . . . . 52
2.10.2. Системная синхронизация . . . . . 56
2.10.3. Схемы сброса . . . . . 58
2.10.4. Прерывания . . . . . 61
2.10.5. Микросхемы памяти . . . . . 63
2.10.6. Затухание сигналов . . . . . 64

**3. Принципы тестирования систем** . . . . . 66
3.1. Программы самоконтроля . . . . . 68
3.2. Существует ли отказ в действительности? . . . 69
3.3. Срок службы интегральных схем . . . . . 71
3.4. Тестирование нагрузками . . . . . 73
3.4.1. Механическая нагрузка . . . . . 74
3.4.2. Температурная нагрузка . . . . . 74
3.4.3. Электрическая нагрузка . . . . . 75
3.5. Локализация отказов . . . . . 75
3.6. Дерево поиска неисправностей . . . . . 79

**4. Применение обычных приборов** . . . . . 82
4.1. Мультиметры . . . . . 83
4.2. Частотомеры . . . . . 84

4.3. Осциллограф . . . . . 85
4.4. Ограниченные возможности обычных приборов . . 90

**5. Ручные инструментальные средства . . . . . 90**

5.1. Логические пробники . . . . . 92
5.1.1. Промышленные логические пробники . . . 97
5.1.2. Использование логического пробника . . . 99
5.2. Логические пульсаторы . . . . . 101
5.2.1. Использование логического пульсатора . . 103
5.3. Тестирование «стимул-реакция» с помощью пульсатора и пробника . . . . . 106
5.4. Индикатор тока . . . . . 107
5.4.1. Использование индикатора тока . . . . 110
5.4.2. Тестирование «стимул-реакция» с помощью логического пульсатора и индикатора тока . . 111
5.5. Логические компараторы . . . . . 116
5.6. Ограниченные возможности ручных инструментальных средств . . . . . 118

**6. Логические анализаторы . . . . . 118**

6.1. Анализаторы логических состояний . . . . . 121
6.2. Анализаторы временных диаграмм . . . . . 134
6.3. Режимы индикации . . . . . 138
6.3.1. Режим карты . . . . . 138
6.3.2. Двоичный и группированный двоичный режим 141
6.3.3. Шестнадцатеричный формат . . . . . 143
6.3.4. Дисассемблерный формат . . . . . 145
6.3.5. Индикация временных диаграмм . . . . 146
6.3.6. Форматы специального кодирования . . . 147
6.4. Дополнительные возможности логических анализаторов 148
6.4.1. Подсчет событий . . . . . 149
6.4.2. Измерения временных интервалов в анализаторе логических состояний . . . . . 151
6.4.3. Диапазоны запуска и условия запуска . . . 153
6.4.4. Предзапуск и постзапуск . . . . . 154
6.4.5. Модули персонификации . . . . . 155
6.4.6. Возможности анализаторов временных диаграмм 156
6.4.7. Принадлежности . . . . . 156

**7. Сигнатурный анализ . . . . . 157**

7.1 Природа цифровых сигналов . . . . . 158
7.2. Счет переходов . . . . . 160
7.3. Вероятность успеха при счете переходов . . . 160
7.4. Коды циклического избыточного контроля . . . 165
7.5. Сигнатурный анализ . . . . . 168
7.6. Вероятность успеха в сигнатурном анализе . . . 170
7.6.1. Обнаружение ошибки при сигнатурном анализе 175
7.7. Простой сигнатурный анализатор . . . . . 177
7.8. Тестирование в режиме свободного счета с использованием сигнатурного анализатора . . . . . 182
7.9. Тестирование ПЗУ в режиме свободного счета . . 187
7.10. Тест-циклы сигнатурного анализа . . . . . 188
7.11. Встроенные средства для сигнатурного анализа . 190

7.12. Ограничения сигнатуриого анализа . . . . . 191
7.13. Сигнатурный анализ как общий способ тестирования 193

**8. Эмуляция** . . . . . 193

8.1. Системы проектирования . . . . . 195
8.1.1. Редактор . . . . . 196
8.1.2. Менеджер файлов . . . . . 198
8.1.3. Редактор связей/загрузчик . . . . . 199
8.1.4. Драйверы устройств ввода-вывода . . . . 199
8.1.5. Ассемблеры . . . . . 200
8.1.6. Компиляторы . . . . . 203
8.1.7. Менеджер памяти . . . . . 204
8.1.8. Отладчик . . . . . 206
8.1.9. Анализатор реального времени . . . . . 207
8.1.10. Программатор ЭИПЗУ . . . . . 207
8.1.11. Внутрисхемные эмуляторы . . . . . 207
8.2. Тестирование систем с применением системы проектирования . . . . . 208
8.3. Автономные эмуляторы . . . . . 209

**9. Программы самоконтроля и диагностики** . . . . . 211

9.1. Программы самоконтроля . . . . . 212
9.1.1. Тестирование ПЗУ . . . . . 213
9.1.2. Тестирование ЗУПВ . . . . . 215
9.2. Диагностические тесты . . . . . 219
9.2.1. Простое тестирование ввода-вывода . . . 220
9.2.2. Инициирование диагиостических тестов . . . 223

**10. Тестирование фуикций ввода-вывода** . . . . . 224

10.1 Пример функционального тестирования подсистемы 225
10.2. Тестированне последовательиых линий связи . . 228
10.2.1. Протоколы последовательной передачи . . 231
10.2.2. Скорость передачи . . . . . 234
10.2.3. Асинхронный связной интерфейс . . . . 235
10.3. Контроль приборной шиины IEEE-488 . . . . 240
10.3.1. Структура универсальной интерфейсной шины 241
10.3.2. Пример передачи по шине . . . . . 246
10.3.3. Анализ сигиалов универсальной интерфейсной шины . . . . . 247

Приложение . . . . . 249

Список литературы . . . . . 250

Производственное издание

**Унльямс Г. Б.**

**ОТЛАДКА МИКРОПРОЦЕССОРНЫХ СИСТЕМ**

Редактор издательства *А. Б. Желдыбин*
Художественный редактор *А. Т. Кирьянов*
Технический редактор *Н. П. Собакина*
Корректор *И. А. Володяева*
ИБ 2179

Сдано в набор 12.03.87. Подписано в печать 13.07.87. Формат 84×108$^{1}/_{32}$. Бумага типографская № 2. Гарнитура литературная. Печать высокая. Усл. печ. л. 13,44 Усл. кр. отт. 13,65. Уч.-изд. л. 13,58. Тираж 50 000 экз. Заказ № 812. Цена 90 к.

Энергоатомиздат, 113114, Москва, М-114, Шлюзовая наб., 10

Владимирская типография Союзполиграфпрома при Госкомиздате СССР
600000, г. Владимир, Октябрьский проспект, д. 7

**Вниманию читателей!**

Энергоатомиздат готовит к изданию в 1988 г. следующие книги:

**Франке К.** Введение в микро-ЭВМ: Пер. с нем. — М.: Энергоатомиздат, 1988 (I). — 8 л.: ил. (В обл.): 55 к., 30 000 экз.

Приведены основные сведения о принципе действия и устройстве ЭВМ, главных элементах и способах обработки информации, принципах и методах связи с внешними устройствами в зависимости от характера и сложности решаемой задачи. Описаны аппаратные и программные средства микро-ЭВМ. Приведены примеры использования микро-ЭВМ в системах реального времени для целей управления, контроля и исследования промышленных систем широкого назначения.

Для инженеров и техников, знакомящихся с возможностями использования микро-ЭВМ в системах управления и контроля. Может быть полезна студентам вузов и учащимся средних учебных заведений.

**Севернс Р., Блум Г.** Импульсные преобразователи постоянного напряжения для систем вторичного электропитания. Пер. с англ. — М.: Энергоатомиздат, 1988 (III). — 22 л.: ил. — (В пер.): 1 р. 80 к., 8000 экз.

Систематически изложены вопросы проектирования мощных ключевых преобразователей постоянного напряжения. Подробно рассмотрены динамические процессы в таких преобразователях с использованием разработанного для этих целей метода усредненного пространственного состояния. Большое внимание уделено перспективным магнитно-связанным преобразователям постоянного напряжения.

Для инженерно-технических работников, занятых в области проектирования источников вторичного электропитания.

**Димитрова М., Пунджев В.** 33 схемы с логическими элементами И—НЕ: Пер. с болг. — Л.: Энергоатомиздат, 1988 (I) — 7 л.: ил. — (В обл.): 50 к., 60 000 экз.

Рассмотрены структуры и параметры основного логического элемента И—НЕ. Даны основы обращения с ним, параметры и характеристики. Описаны 33 характерные схемы — логические и импульсные. Даются описания для их изготовления и использования, объяснено действие, перечислены условия выбора элементов. Даны сведения по измерению режимов в различных точках. Описаны устройства, облегчающие работу со схемами, и средства контроля.

Для инженерно-технических работников, занимающихся электроникой, и студентов вузов.

**Боон К. Л. ПАСКАЛЬ** для всех: Пер. с голланд. — М.: Энергоатомиздат, 1988 (II). — 9 л.: ил. — (В обл.): 60 к., 50 000 экз.

В доступной форме изложены основы программирования на языке ПАСКАЛЬ. Хотя книга ориентирована на читателей, не имеющих опыта работы с вычислительной техникой, она может быть полезна и специалистам, так как позволяет не только познакомиться с языком, но и по-новому взглянуть на известные принципы программирования.

Для инженерно-технических работников, студентов и школьников.

Приобрести эти книги Вы сможете во всех магазинах, распространяющих научно-техническую литературу.

90 к.

ЭНЕРГОАТОМИЗДАТ